

# UNDERSTANDING CITYSCAPES

Efficient Urban Semantic Scene Understanding

---

Dissertation approved by

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik

for the degree of

Doktor-Ingenieur (Dr.-Ing.)

by

MARIUS CORDTS

Master of Science (M.Sc.)

born in Aachen, Germany

---

Examiner: Prof. Stefan Roth, Ph.D.

Co-examiner: Prof. Dr. Bernt Schiele

Date of Submission: September 4<sup>th</sup>, 2017

Date of Defense: October 17<sup>th</sup>, 2017

---

Darmstadt, 2017

D17

---



## ABSTRACT

---

Semantic scene understanding plays a prominent role in the environment perception of autonomous vehicles. The car needs to be aware of the semantics of its surroundings. In particular it needs to sense other vehicles, bicycles, or pedestrians in order to predict their behavior. Knowledge of the drivable space is required for safe navigation and landmarks, such as poles, or static infrastructure such as buildings, form the basis for precise localization. In this work, we focus on visual scene understanding since cameras offer great potential for perceiving semantics while being comparably cheap; we also focus on urban scenarios as fully autonomous vehicles are expected to appear first in inner-city traffic. However, this task also comes with significant challenges. While images are rich in information, the semantics are not readily available and need to be extracted by means of computer vision, typically via machine learning methods. Furthermore, modern cameras have high resolution sensors as needed for high sensing ranges. As a consequence, large amounts of data need to be processed, while the processing simultaneously requires real-time speeds with low latency. In addition, the resulting semantic environment representation needs to be compressed to allow for fast transmission and down-stream processing. Additional challenges for the perception system arise from the scene type as urban scenes are typically highly cluttered, containing many objects at various scales that are often significantly occluded.

In this dissertation, we address efficient urban semantic scene understanding for autonomous driving under three major perspectives. First, we start with an analysis of the potential of exploiting multiple input modalities, such as depth, motion, or object detectors, for semantic labeling as these cues are typically available in autonomous vehicles. Our goal is to integrate such data holistically throughout all processing stages and we show that our system outperforms comparable baseline methods, which confirms the value of multiple input modalities. Second, we aim to leverage modern deep learning methods requiring large amounts of supervised training data for street scene understanding. Therefore, we introduce *Cityscapes*, the first large-scale dataset and benchmark for urban scene understanding in terms of pixel- and instance-level semantic labeling. Based on this work, we compare various deep learning methods in terms of their performance on inner-city scenarios facing the challenges introduced above. Leveraging these insights, we combine suitable methods to obtain a real-time capable neural network for pixel-level semantic labeling with high classification accuracy. Third, we combine our previ-

ous results and aim for an integration of depth data from stereo vision and semantic information from deep learning methods by means of the *Stixel World* (Pfeiffer and Franke, 2011b). To this end, we reformulate the *Stixel World* as a graphical model that provides a clear formalism, based on which we extend the formulation to multiple input modalities. We obtain a compact representation of the environment at real-time speeds that carries semantic as well as 3D information.

## ZUSAMMENFASSUNG

---

Für die Umgebungserfassung autonomer Fahrzeuge ist das semantische Szenenverständnis von großer Bedeutung. Das autonome Fahrzeug muss seine Umgebung wahrnehmen und verstehen können. Insbesondere müssen andere Fahrzeuge, Fahrräder oder Fußgänger erkannt werden, um ihr Verhalten zu prädictieren. Die Basis für sichere Navigation ist ein exaktes Wissen über die befahrbare Umgebung, während eine präzise Lokalisierung mittels Landmarken (z.B. Pfeiler) oder statischer Infrastruktur (z.B. Gebäude) ermöglicht wird. Da Kameras ein großes Potenzial für die Wahrnehmung von Semantik bieten und zudem vergleichsweise günstig sind, liegt der Fokus dieser Arbeit auf dem visuellen Verstehen von Szenen. Ein weiterer Fokus liegt auf dem urbanen Umfeld, da vollautonome Fahrzeuge zuerst im innerstädtischen Verkehr erwartet werden. Nichtsdestotrotz birgt diese Aufgabe auch signifikante Herausforderungen. Obwohl Bilder informationsreich sind, ist die Semantik nicht unmittelbar verfügbar und muss zunächst durch Methoden der Bildverarbeitung, typischerweise mittels Verfahren des maschinellen Lernens, extrahiert werden. Des Weiteren haben moderne Kameras hochauflösende Sensoren, um hohe Erkennungsreichweiten zu erreichen. Als Konsequenz daraus müssen große Datenmengen verarbeitet werden, was in Echtzeit bei niedriger Latenz geschehen muss. Zusätzlich muss die resultierende semantische Umgebungsrepräsentation komprimiert werden, um eine schnelle Übertragung und Weiterverarbeitung zu ermöglichen. Weitere Herausforderungen für das System zur Umgebungserfassung ergeben sich durch den Szenentyp, da urbane Szenen typischerweise unübersichtlich sind und viele Objekte in verschiedenster Größe und mit signifikanten Verdeckungen beinhalten.

In dieser Dissertation wird effizientes urbanes semantisches Szenenverstehen für autonomes Fahren unter drei Hauptgesichtspunkten adressiert. Erstens werden die Möglichkeiten hinter der Benutzung von mehreren Eingangsmodalitäten, wie zum Beispiel Tiefe, Bewegung oder Objektdetektion, für semantisches Labeln analysiert, da diese Informationen typischerweise in autonomen Fahrzeugen verfügbar sind. Hierbei ist das Ziel, diese Daten vollständig und durchgängig in alle Verarbeitungsschritte zu integrieren, mit dem Ergebnis, dass das System die Performance von Vergleichsmethoden übertrifft, was den Wert von mehreren Inputmodalitäten bestätigt. Zweitens wird darauf abgezielt moderne Methoden aus dem Bereich Deep Learning, die große annotierte Trainingsdatenmengen benötigen, für das Verstehen von Straßenszenen einzusetzen. Dazu wird *Cityscapes* eingeführt, der erste großangelegte Datensatz und Benchmark für urbanes Szenenverstehen mittels semantischem Labeln auf Pixel- und

Instanzebene. Basierend auf dieser Arbeit werden verschiedene Methoden des Deep Learnings hinsichtlich ihrer Performance auf Innenstadtszenarien bezüglich obiger Herausforderungen verglichen. Auf Basis dieser Erkenntnisse werden geeignete Methoden kombiniert, um ein echtzeitfähiges neuronales Netz für semantisches Labeln auf Pixel-Ebene mit hoher Klassifikationsgenauigkeit zu erhalten. Drittens werden die vorigen Ergebnisse mit dem Ziel kombiniert, Tiefendaten aus Stereobildverarbeitung und semantische Informationen von Deep Learning-Methoden mit Hilfe der *Stixel Welt* (Pfeiffer und Franke, 2011b) zu integrieren. Zu diesem Zweck wird die *Stixel Welt* als graphisches Modell umformuliert, so dass ein klarer Formalismus existiert, auf Basis dessen das Modell auf mehrere Eingangsmodalitäten erweitert wird. Resultierend ergibt sich eine kompakte Repräsentation der Umgebung, die in Echtzeit berechnet werden kann und semantische sowie 3D Informationen beinhaltet.

## ACKNOWLEDGMENTS

---

First and foremost, I would like to express my gratitude to my advisor Prof. Stefan Roth for accepting me as a Ph.D. student in his Visual Inference Group at TU Darmstadt. Thank you for all the valuable input to my research, your dedication in the days and nights before submission deadlines, and for supporting my development as a research scientist. I am especially grateful for such an excellent level of supervision and an integration into the research group given that I am an external Ph.D. candidate. In particular, the retreats allowed for a valuable exchange with other Ph.D. students and researchers in related fields and supported the creation of ideas and collaborations.

Next, I would like to thank Dr. Uwe Franke for providing me with the opportunity to conduct my research in his Image Understanding Group at Daimler R&D. You created an excellent research environment in which it is a great pleasure to work, to experiment, and to discuss newest ideas. You preserve the research atmosphere despite structural changes and in doing so allow for continuous and focused research. Thank you for all the valuable impulses regarding my work, your trust, and your support of my career.

My sincere and deepest gratitude goes to Dr. Markus Enzweiler, my technical advisor at Daimler R&D. You were my mentor in many aspects throughout the Ph.D. studies and beyond. You were always encouraging and inspiring, you had an open ear for my ideas and problems, and you provided continuous feedback and insightful comments for my work.

My thanks also go to the remaining colleagues from the Image Understanding Group, especially to my closest collaborators Timo Rehfeld and Lukas Schneider. Thank you for all the valuable discussions and the hard questions that allowed me to dive deeper into my research problems and to widen my horizon. Thank you for creating a great atmosphere that makes our working hours and beyond truly enjoyable. I found many long-term friends in you. My warmest gratitude also goes to Nick Schneider, who has become a very close friend during my Ph.D. studies.

I would like to thank my parents Waltraud and Harald as well as my sister Isabell. You always guide me in my decisions and choices, you are there when times are rough, and you are the biggest support in my life. Eventually, thank you to Ann-Kathrin, for your endless support when working on this dissertation, for joining me during numerous recordings at weekends, and for all your love.





## CONTENTS

---

1	INTRODUCTION	1
1.1	Digital Cameras	1
1.2	Semantic Scene Understanding	3
1.3	Computer Vision in Automobiles	6
1.4	Dissertation Goals	9
1.5	Dissertation Outline	10
1.5.1	Contributions	10
2	RELATED WORK	13
2.1	Overview of Pixel-level Semantic Labeling	13
2.1.1	Classic processing pipeline	13
2.1.2	Deep learning	15
2.2	Constrained Scene Type	17
3	MULTI-CUE SEMANTIC LABELING	19
3.1	Introduction	20
3.2	Encode-and-Classify Trees	22
3.2.1	Pixel-level semantic labeling	24
3.2.2	Texton maps	25
3.2.3	Combination	26
3.2.4	Training	26
3.3	Superpixels	27
3.3.1	Incorporating depth	28
3.3.2	Incorporating pixel classification	28
3.4	Multi-cue Segmentation Tree	29
3.4.1	Proximity cues	30
3.4.2	Tree construction	34
3.5	Multi-cue Region Classification	35
3.5.1	Multi-cue features	35
3.5.2	Training	37
3.6	CRF Inference	38
3.6.1	Unaries	39
3.6.2	Parent-child	39
3.6.3	Inference	40
3.7	Temporal Regularization	40
3.8	Experiments	41
3.8.1	Cues	41
3.8.2	Datasets	42
3.8.3	Metrics	42
3.8.4	Comparison to baselines	45
3.8.5	Ablation studies	49
3.8.6	Qualitative results	50

3.8.7	Run time . . . . .	50
3.9	Discussion . . . . .	53
4	LARGE-SCALE SEMANTIC SCENE UNDERSTANDING	55
4.1	Introduction . . . . .	57
4.2	Dataset . . . . .	58
4.2.1	Data specifications . . . . .	59
4.2.2	Classes and annotations . . . . .	61
4.2.3	Dataset splits . . . . .	64
4.2.4	Statistical analysis . . . . .	66
4.3	Pixel-level Semantic Labeling . . . . .	72
4.3.1	Metrics . . . . .	73
4.3.2	Control experiments . . . . .	74
4.3.3	Baselines . . . . .	75
4.3.4	Pixel-level benchmark . . . . .	81
4.3.5	Real-time fully convolutional networks . . . . .	94
4.3.6	Cross-dataset evaluation . . . . .	98
4.4	Instance-level Semantic Labeling . . . . .	99
4.4.1	Metrics . . . . .	99
4.5	Benchmark Suite . . . . .	101
4.6	Impact and Discussion . . . . .	102
5	SEMANTIC STIXELS	107
5.1	Related Scene Representations . . . . .	110
5.1.1	Unsupervised image segmentation . . . . .	110
5.1.2	Street scene environment models . . . . .	111
5.1.3	Layered environment models . . . . .	111
5.2	The Stixel Model . . . . .	112
5.3	Graphical Model . . . . .	113
5.3.1	Prior . . . . .	115
5.3.2	Data likelihood . . . . .	117
5.4	Inference . . . . .	120
5.4.1	Dynamic programming . . . . .	121
5.4.2	Algorithmic simplification . . . . .	122
5.4.3	Approximations . . . . .	123
5.5	Parameter Learning . . . . .	124
5.6	Object-level Knowledge . . . . .	125
5.6.1	Extended Stixel model . . . . .	126
5.7	Experiments . . . . .	129
5.7.1	Datasets . . . . .	131
5.7.2	Metrics . . . . .	131
5.7.3	Baseline . . . . .	132
5.7.4	Impact of input cues . . . . .	132
5.7.5	Impact of approximations . . . . .	135
5.8	Discussion . . . . .	136
6	DISCUSSION AND OUTLOOK	137

6.1	Discussion . . . . .	137
6.2	Future Work . . . . .	139
A	APPENDIX . . . . .	143
A.1	Cityscapes Label Definitions . . . . .	143
A.1.1	Human . . . . .	143
A.1.2	Vehicle . . . . .	144
A.1.3	Nature . . . . .	145
A.1.4	Construction . . . . .	145
A.1.5	Object . . . . .	146
A.1.6	Sky . . . . .	147
A.1.7	Flat . . . . .	147
A.1.8	Void . . . . .	148
	BIBLIOGRAPHY . . . . .	149

## LIST OF FIGURES

Figure 1.1	Examples for scene recognition and image classification . . . . .	4
Figure 1.2	Examples for object detection and semantic labeling . . . . .	5
Figure 1.3	Examples complex urban scenes . . . . .	8
Figure 3.1	Multi-cue system overview . . . . .	21
Figure 3.2	Encode-and-classify tree structure . . . . .	23
Figure 3.3	Superpixel variants for region proposal tree . . . . .	29
Figure 3.4	Segmentation tree and resulting CRF model . . . . .	35
Figure 3.5	Overlapping segments of ground truth and prediction . . . . .	44
Figure 3.6	Overview of different annotation and prediction formats . . . . .	46
Figure 3.7	Qualitative results on DUS dataset . . . . .	51
Figure 3.8	Breakdown of run time . . . . .	52
Figure 4.1	Examples Cityscapes dataset . . . . .	56
Figure 4.2	Preprocessing variants of HDR images . . . . .	60
Figure 4.3	Number of annotated pixels . . . . .	62
Figure 4.4	Exemplary labeling process . . . . .	63
Figure 4.5	Annotation examples at extremes of Cityscapes dataset . . . . .	65
Figure 4.6	Proportions of annotated pixels . . . . .	70
Figure 4.7	Dataset statistics regarding scene complexity . . . . .	72
Figure 4.8	Histogram of vehicle distances . . . . .	72
Figure 4.9	Control experiments: most instances . . . . .	78
Figure 4.10	Control experiments: most cars . . . . .	79
Figure 4.11	Baseline experiments: most instances . . . . .	83
Figure 4.12	Baseline experiments: most cars . . . . .	83
Figure 4.13	Benchmarked methods: most instances . . . . .	89
Figure 4.14	Benchmarked methods: most cars . . . . .	90
Figure 4.15	Histogram over number of correct methods . . . . .	93
Figure 4.16	Activity statistics for Cityscapes dataset over time . . . . .	103
Figure 4.17	Best performing methods on Cityscapes dataset over time . . . . .	104
Figure 5.1	Semantic Stixels' scene representation . . . . .	109
Figure 5.2	Stixel world as a factor graph . . . . .	114
Figure 5.3	Disparity measurements and resulting Stixel segmentation . . . . .	119
Figure 5.4	Shortest path problem . . . . .	124
Figure 5.5	Examples of object-level knowledge . . . . .	126

Figure 5.6	Stixel model with object-level knowledge . . .	127
Figure 5.7	Exemplary results of our Semantic Stixel model on Cityscapes . . . . .	130
Figure 5.8	Qualitative results of ablation studies . . . . .	133
Figure 5.9	Analysis of influence of semantic input channel	135

## LIST OF TABLES

---

Table 3.1	Features for segmentation tree construction . .	31
Table 3.2	Features for multi-cue region classification . .	37
Table 3.3	Binary confusion matrix. . . . .	43
Table 3.4	Averaged quantitative results on DUS dataset	47
Table 3.5	Averaged quantitative results on KITTI dataset	48
Table 3.6	Detailed quantitative results on DUS dataset .	49
Table 3.7	Detailed quantitative results on KITTI dataset	50
Table 4.1	Evaluation of FCN model on different subsets	67
Table 4.2	Evaluation of FCN models on temperature ex- tremes . . . . .	67
Table 4.3	Comparison of Cityscapes to related datasets .	68
Table 4.4	Absolute number and density of annotated pix- els . . . . .	69
Table 4.5	Absolute and average number of instances . .	71
Table 4.6	Control experiments: averages . . . . .	75
Table 4.7	Control experiments: classes p1 . . . . .	76
Table 4.8	Control experiments: classes p2 . . . . .	77
Table 4.9	Baseline experiments: averages . . . . .	81
Table 4.10	Baseline experiments: classes p1 . . . . .	82
Table 4.11	Baseline experiments: classes p2 . . . . .	82
Table 4.12	Benchmarked methods: overview . . . . .	85
Table 4.13	Benchmarked methods: averages . . . . .	86
Table 4.14	Benchmarked methods: classes p1 . . . . .	87
Table 4.15	Benchmarked methods: classes p2 . . . . .	88
Table 4.16	Comparison of networks in terms of classifica- tion performance and run time . . . . .	96
Table 4.17	Quantitative results of FCN variants based on GoogLeNet . . . . .	97
Table 4.18	Quantitative results on CamVid and KITTI ob- tained by Cityscapes model . . . . .	99
Table 5.1	Performance of Semantic Stixels with different input combinations . . . . .	134

Table 5.2	Performance of approximated inference . . . .	136
-----------	---	-----

## LIST OF ALGORITHMS

---

3.1	Segmentation tree construction . . . . .	34
3.2	Generate object instances from label prediction . . . . .	48

## ACRONYMS

---

ADAS	Advanced Driver Assistance Systems
AP	Average Precision
BoF	Bag-of-Features
BoW	Bag-of-Words
CamVid	Cambridge-driving Labeled Video Database (dataset, Brostow et al., 2009)
CMOS	Complementary Metal-Oxide-Semiconductor
CNN	Convolutional Neural Network
COCO	Common Objects in Context (dataset, T.-Y. Lin et al., 2014)
CPU	Central Processing Unit
CRF	Conditional Random Field
CVPR	IEEE Conference on Computer Vision and Pattern Recognition
DAG	Directed Acyclic Graph
DNN	Deep Neural Network
DP	Dynamic Programming
DSLR	Digital Single-Lens Reflex
DUS	Daimler Urban Segmentation (dataset, Scharwächter et al., 2013)

ERC	Extremely Randomized Clustering (Moosmann et al., 2008)
FCN	Fully Convolutional Network (Shelhamer et al., 2017)
FN	False Negative
FPGA	Field-Programmable Gate Array
FP	False Positive
GBIS	Graph Based Image Segmentation (Felzenszwalb and Huttenlocher, 2004)
gPb	Globalized Probability of Boundary (Maire et al., 2008)
GPS	Global Positioning System
GPU	Graphics Processing Unit
GT	Ground Truth
HDR	High Dynamic-Range
HIK	Histogram Intersection Kernel (J. Wu, 2010)
HMM	Hidden Markov Model
ICCV	International Conference on Computer Vision
iFN	Instance False Negative
iIoU	Instance Intersection-over-Union
ILSVRC	ImageNet Large Scale Visual Recognition Challenge (Russakovsky et al., 2015)
IoU	Intersection-over-Union
iTP	Instance True Positive
JI	Jaccard Index
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago (dataset, Geiger et al., 2013)
KLT	Kanade-Lucas-Tomasi (feature tracker, Tomasi and Kanade, 1991)
LAB	Lightness L, color-opponents A and B (color space)
LDR	Low Dynamic-Range
MAP	maximum-a-posteriori

MRF	Markov Random Field
NYUD <sub>2</sub>	New York University Depth, Version 2 (dataset, Silberman et al., 2012)
OWT	Oriented Watershed Transform
PASCAL	Pattern Analysis, Statistical Modelling and Computational Learning
PASCAL VOC	PASCAL Visual Object Classes (dataset, Everingham et al., 2014)
R-CNN	Region-based Convolutional Neural Network (Girshick et al., 2014)
RNN	Recurrent Neural Network
RDF	Randomized Decision Forest
RGB-D	Red Green Blue Depth (color space with depth data)
RGB	Red Green Blue (color space)
S-SVM	Structured Support Vector Machine (Nowozin and Lampert, 2011)
SEOF	Superpixels in an Energy Optimization Framework (Veksler et al., 2010)
SGD	Stochastic Gradient Descent
SGM	Semi-Global Matching
SIFT	Scale-Invariant Feature Transform
SLIC	Simple Linear Iterative Clustering (superpixels, Achanta et al., 2012)
SPC	Static Predictor Coarse
SPF	Static Predictor Fine
SUN	Scene UNderstanding (dataset, Xiao et al., 2016)
SUV	Suburban Utility Vehicle
SVM	Support Vector Machine
TN	True Negative
TPR	True Positive Rate
TP	True Positive
UCM	Ultrametric Contour Map (Arbeláez et al., 2011)



VGG Visual Geometry Group (used as acronym for [CNN](#) architectures proposed by Simonyan and Zisserman, [2015](#))



# INTRODUCTION

---

## CONTENTS

---

1.1	Digital Cameras . . . . .	1
1.2	Semantic Scene Understanding . . . . .	3
1.3	Computer Vision in Automobiles . . . . .	6
1.4	Dissertation Goals . . . . .	9
1.5	Dissertation Outline . . . . .	10
1.5.1	Contributions . . . . .	10

---

In this dissertation, we will investigate *semantic scene understanding* via image analysis with a focus on *pixel-level semantic labeling*. In doing so, we will target *automotive* applications and address *urban* scenes in order to work towards fully *autonomous driving*. In this chapter, we motivate this research topic and discuss the goals of the dissertation at hand. To this end, we start with an introduction of digital cameras and their impact on modern society in Section 1.1. We continue with an overview of the major tasks for semantic scene understanding (Section 1.2) and then turn to applications in the automotive domain (Section 1.3). Eventually, the goals of the dissertation and its outline are provided in Section 1.4 and Section 1.5, respectively.

## 1.1 DIGITAL CAMERAS

The birth of practically and commercially usable photography dates back to a public presentation of Louis-Jacques-Mandé Daguerre in 1839 (Daniel, 2004). Ever since, people are fascinated by photography and take pictures, e.g. for documentation, preserving memories, or as artwork. Images best reflect the way that humans perceive the environment, i.e. mainly with their eyes, and are rich in detail and contained information. The next major development step of cameras was achieved with the commercial availability of digital cameras around 1990 (Larish, 1990; Said, 1990). These cameras superseded chemical solutions requiring photographic films, are cheaper and faster to use, and allow for digital post-processing, storing, and sharing of images. The third major impact was created with the availability of cameras in mobile phones in 2000 (Hill, 2013) leading to today's smartphones combining phone and computer functionality in a mobile internet capable device.

In parallel to the spreading of digital cameras, they underwent a rapid technical development. The image quality improved due to better sensor technology, resolution, and optics. Further, there is more and more image processing available within the camera device, such as High Dynamic-Range (HDR) photography or panorama stitching. Eventually, a multitude of online services became available, where images can be uploaded, post-processed, stored, shared, or searched. These services can be categorized into different groups depending on their primary application such as image hosting websites, e.g. Flickr<sup>1</sup> or Instagram<sup>2</sup>, social networks, e.g. Facebook<sup>3</sup> or Google+<sup>4</sup>, image search engines, e.g. provided by Google<sup>5</sup> or Microsoft<sup>6</sup>, and instant messaging services, e.g. WhatsApp<sup>7</sup> or Snapchat<sup>8</sup>. In addition, there exist platforms primarily for video content, e.g. YouTube<sup>9</sup>.

Boosted by the rapid spreading of digital cameras and smartphones, there is an exploding number of images taken per day. The number of images shared via online services is exponentially increasing in recent years (Meeker, 2016). In 2015, the daily number of images shared on Facebook was estimated as 500 million and the number of images sent via the instant messengers WhatsApp, Facebook Messenger, and Snapchat reached almost three billion per day (Meeker, 2016). In the same year, 400 hours of video material were uploaded on YouTube per minute (Jarboe, 2015). Already impressive, these numbers reflect only a lower bound on the number of pictures and videos taken per day as they do not count other online services, data that is stored offline, or data that is not stored at all, e.g. from surveillance cameras.

These numbers can only indicate the importance of images for our every-day life and their impact on our society. Especially, from a commercial perspective, there is a huge interest in automatically parsing and understanding the large amount of information contained in images. The intention of online services is to gain knowledge about the users and to provide services such as image search or person identification in order to attract users. The major goal in surveillance is to understand the observed scene and to detect anomalies. In robotics and autonomous driving, cameras are often the eyes of the robots and their images need to be analyzed in order to navigate safely within the environment.

While being an exciting task, automatic image understanding is also a hard problem. If the online services introduced above seek to parse all the images that are uploaded every day, they need to deal

---

<sup>1</sup> [www.flickr.com](http://www.flickr.com)

<sup>2</sup> [www.instagram.com](http://www.instagram.com)

<sup>3</sup> [www.facebook.com](http://www.facebook.com)

<sup>4</sup> [plus.google.com](http://plus.google.com)

<sup>5</sup> [images.google.com](http://images.google.com)

<sup>6</sup> [www.bing.com/images](http://www.bing.com/images)

<sup>7</sup> [www.whatsapp.com](http://www.whatsapp.com)

<sup>8</sup> [www.snapchat.com](http://www.snapchat.com)

<sup>9</sup> [www.youtube.com](http://www.youtube.com)

with a big amount of data. A single image consists of several million values that need to be processed, imposing high computational demands and requiring efficient methods. Simultaneously, while digital images are easily accessible for computers, the sensing, however, imitates the human visual perception. Thus, the contained information is not readily available and needs to be extracted via computer vision techniques. Due its high interest paired with these challenges, image understanding plays a major role in modern computer vision research.

## 1.2 SEMANTIC SCENE UNDERSTANDING

The field of semantic scene understanding addresses automated image understanding from a semantic perspective, i.e. parsing the image and assigning a meaning to it or its parts. Such a meaning is typically encoded by a class label, e.g. person, building, or car, while the set of possible class labels is usually defined beforehand. Nowadays, virtually all methods for semantic scene parsing use machine learning techniques (Everingham et al., 2014; T.-Y. Lin et al., 2014; Russakovsky et al., 2015). Note that scene understanding extends to videos as well, since videos are merely a collection of temporally successive frames. These frames can either be processed as individual images or scene understanding can be improved by exploiting temporal information. In recent years, there has been a rapid progress in this field and many of the above mentioned online services exploit scene understanding for their products. In the following, we provide a brief overview of major tasks of semantic scene understanding. All these tasks are driven by corresponding datasets for training and evaluating of algorithms. These datasets are often paired with benchmarks such that different methods can be directly compared. In addition to the introduced tasks, there exist hybrid or combined variants, or rather specialized tasks such as human keypoint detection or fine-grained classification.

**SCENE RECOGNITION** The task of scene recognition aims at determining the scene type that an image was taken in, c.f. Figure 1.1 for example images and ground truth scene types. Note that in the example in Figure 1.1b, the task at hand is to predict the background scenery, i.e. *ocean*, and not the predominant object in the image, i.e. *boat*. However, as the example indicates, scene type and contained objects are often closely correlated and thus is their recognition. Popular datasets and benchmarks for scene recognition are Scene Understanding (SUN; Xiao et al., 2016) and Places (Zhou et al., 2016).

**IMAGE CLASSIFICATION** Compared with scene recognition, the roles in image classification are essentially flipped and the task is

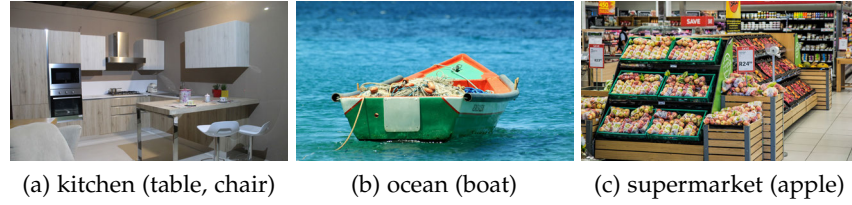


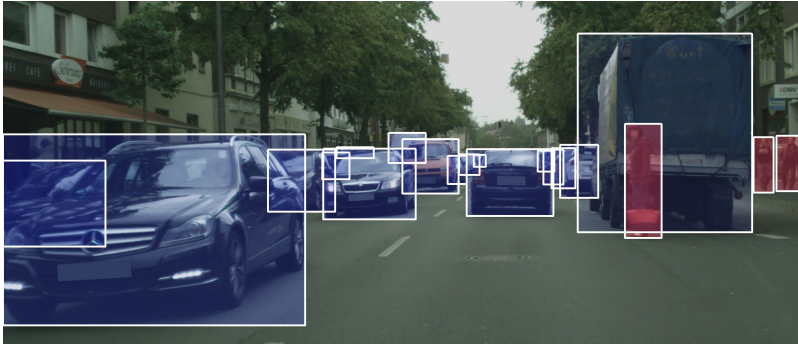
Figure 1.1: Example images and ground truth annotations for a typical scene recognition (image classification) task.

to classify the dominant objects in the image, c.f. Figure 1.1 and the ground truth labels in brackets. The most popular dataset for image classification is ImageNet (Russakovsky et al., 2015) paired with the annual ImageNet Large Scale Visual Recognition Challenges (ILSVRCs).

**ACTIVITY RECOGNITION** While scene recognition and image classification focus on the scene type and the foreground objects, respectively, activity recognition aims to determine the ongoing activities in the image. Note that since activities are typically most prominent in the temporal domain, this task is typically applied to videos. There exist several datasets for activity recognition that have a focus on different action types, e.g. Charades (Sigurdsson et al., 2016) for everyday life activities, ActivityNet (Heilbron et al., 2015) containing human activities, or Sports-1M (Karpathy et al., 2014) with sport activities.

**BOUNDING BOX OBJECT DETECTION** The three previously introduced tasks all focus on the image/video level, i.e. the output describes the scene (or a temporal subset) as a whole. With bounding box object detection the task is not only to classify the objects within a scene, but also to localize them in the image. Thus, the ideal output of a system for this task is a set of rectangular axially parallel boxes that form the smallest bounding box around the object of interest associated with a class label and often a confidence score. Note that depending on the context, the term *bounding box* can be omitted in the task's name. Popular datasets and benchmarks for object detection are ILSVRC (Russakovsky et al., 2015), Microsoft Common Objects in Context (COCO; T.-Y. Lin et al., 2014), or KITTI (Geiger et al., 2013). An example image with object bounding boxes is provided in Figure 1.2a.

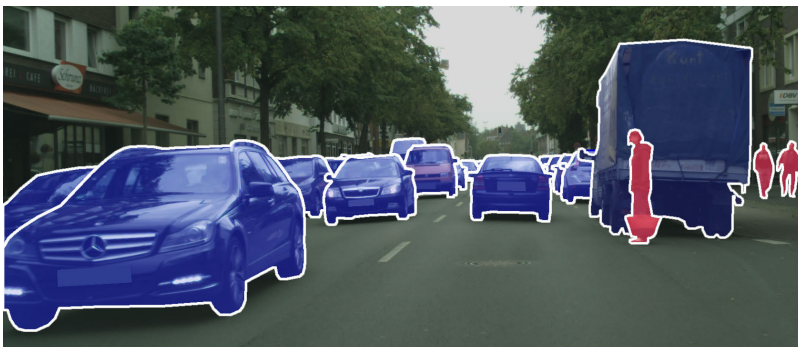
Figure 1.2a also unveils that object detection has three potential shortcomings in representing the scene. First, some classes such as road or sky are not countable and thus their individual instances are not well defined. Second, elongated objects such as buildings or trees are poorly described by their bounding boxes. Third, as evident from



(a) Bounding Box Object Detection



(b) Pixel-level Semantic Labeling



(c) Instance-level Semantic Labeling

Figure 1.2: Example images and ground truth annotations for bounding box object detection, pixel-level semantic labeling, and instance-level semantic labeling. Note that object detection separates individual instances, but does not provide any segmentation and is only applicable to compact, non-elongated objects. Pixel-level semantic labeling, in turn, is instance agnostic, but segments arbitrary object shapes. Instance-level semantic labeling aims to combine the two other tasks. The image is taken from the Cityscapes dataset as introduced in Chapter 4 and classes are encoded by false colors, e.g. vehicles are colored in blue, humans in red, etc.



the cars in the left part of Figure 1.2a, bounding boxes for heavily occluded objects overlap and cause ambiguities in their representation.

**PIXEL-LEVEL SEMANTIC LABELING** The task behind pixel-level semantic labeling is to assign a semantic class label to each pixel of the image, c.f. Figure 1.2b. In doing so, the limitations of object detection are circumvented and the labeling can be applied to classes of arbitrary shape. Alternatively this task is also denoted as *scene labeling* or *semantic segmentation*, where the latter emphasizes that image regions belonging to the same semantic class are segmented. Note however that this task does not separate individual instances anymore, instead all instances of the same class are assigned to the same label. As a consequence, the cars in Figure 1.2b are all grouped in the same segment. Prominent datasets for this task with corresponding benchmark are [PASCAL VOC](#) (Everingham et al., 2014), [PASCAL-Context](#) (Mottaghi et al., 2014), Microsoft [COCO](#) (T.-Y. Lin et al., 2014), and [SUN RGB-D](#) (Song et al., 2015). Note that a more detailed comparison and discussion of these datasets follows in Chapter 4.

**INSTANCE-LEVEL SEMANTIC LABELING** Instance-level semantic labeling extends the pixel-level task by simultaneously assigning each pixel to a semantic label and an object instance, c.f. Figure 1.2c. In doing so, the representational power of the two previous tasks is combined. Since an instance segmentation for uncountable classes is not well defined, these classes are typically omitted, c.f. Figure 1.2c. Nevertheless, a combination with a pixel-level semantic labeling for such classes is straight-forward. Benchmark datasets for this task are [PASCAL VOC](#) (Everingham et al., 2014) and Microsoft [COCO](#) (T.-Y. Lin et al., 2014).

### 1.3 COMPUTER VISION IN AUTOMOBILES

In parallel with the advances of camera technology, the development of the modern automobile started with the patent *vehicle with gas engine operation* by Benz, 1886. Ever since automobiles quickly spread over the world, reaching a global production of 91 million units in 2015 (OICA, 2016). However, the wide dissemination of automobiles comes at a high cost. For 2013, the number of road accident fatalities is estimated as 1.25 million being the main cause of death among people aged 15 to 29 years (WHO, 2015). As a consequence, legislation, researchers, and car industry continuously aim to improve the automobile’s safety via passive and active safety systems. Passive safety includes systems such as airbags, seatbelts, or the vehicle’s physical structure, to protect occupants and road users during a crash. Active safety aims to prevent the crash at all and, amongst others, leverages cameras equipped with computer vision to sense the environment.



Such systems are typically denoted as Advanced Driver Assistance Systems (ADAS) and aim to increase the car and road safety by supporting the driving process. Examples for such components are lane detection and departure warning systems, pedestrian detection systems with autonomous breaking, or traffic sign recognition. These examples also indicate the importance of a semantic understanding and show that object detection systems are already available in actual products.

With increasing advances in technology, the vehicle is capable of conducting more and more driving tasks itself. Accordingly, the report of SAE International, 2014 defines six levels of autonomy ranging from *no automation* to *full automation*, where the latter stands for a vehicle that is capable of driving safely without driver in all circumstances that can be managed by a human driver. In other words, a fully autonomous vehicle matches, or exceeds, human driving performance. The motivation of researchers and industry to push forward such technology is to reach a level of maximum safety while being able to offer services such as *robotaxis* that can pickup and drive passengers. For such systems, computer vision plays an important role in sensing the environment with cameras and ultimately constituting the eye of the vehicle.

For ADAS and even more for autonomous driving, a semantic scene understanding as described in Section 1.2 plays a major role. The vehicle needs to sense its environment and needs to know about the scene content for safe navigation and interaction with the environment. In order to achieve a 360° coverage around the vehicle paired with a sufficient level of redundancy, often multiple sensors out of different technologies are combined, such as radar sensors, laser scanners, or cameras. Each sensor has its individual strengths; cameras are typically the cheapest, come closest to human perception, and allow for a recognition of many semantic classes.

While scene type, the set of contained classes, and high-level actions typically vary little when considering street scenes, the location of objects and scene elements is highly important. In order to accurately predict the behavior of active road users, e.g. cars, trucks, pedestrians, or cyclists, the autonomous vehicle needs to classify and localize them. Further, it needs to be aware of the drivable space and should be capable of segmenting different ground regions, e.g. sidewalks in order to anticipate occluded pedestrians. Further, landmarks such as poles, walls, or fences aid a precise localization of the autonomous vehicle within a map of its environment. Since buildings have a nearly constant appearance during all seasons of the year, they are well suited for a feature-based localization, whereas trees are not. Traffic lights and traffic signs need to be recognized in order to obey traffic regulations. Even the sky is an important image region, since depth estimations via stereo vision are typically very noisy in such



(a) High Variations in Scale



(b) Severe Occlusions

Figure 1.3: Example images for complex, cluttered, and busy urban scenes. Note the challenges that arise from large variations in scale, e.g. the close and far cars in (a). Further note the occluded pedestrians to the right of the ego vehicle in (b) that are about to enter the road.

texture-less regions. Overall, these requirements emphasize the need for a high-performing semantic scene understanding system.

Facing an automotive environment, such a scene understanding system is confronted with several challenges. Particularly in an urban environment where autonomous vehicles are extensively developed, the scenes are highly unstructured and cluttered. Objects can occur at various scales and there are often many objects in a scene that need to be detected despite severe occlusions, c.f. Figure 1.3. In addition to the challenges stemming from an urban environment, an appropriate system should reliably work under all circumstances and thus must be robust against common effects in the automotive domain, e.g. pixel noise, motion blur, illumination changes, varying weather conditions, or wiper occlusions. Simultaneously, such a system must be efficient and capable of parsing the scene at camera frame rates, typically around 20 Hz, with limited computational resources and power budget. Eventually, the output of a perception system is often sent to a downstream processing stage such as sensor fusion or behavior planning. Therefore, the scene representation must be compact, i.e. low bandwidth, and easily parsable for the downstream algorithm. These challenges paired with the goal of reaching a new level of autonomy and safety motivate many researchers to work in this domain and also form the foundations of this work.

## 1.4 DISSERTATION GOALS

In the previous sections, we learned that semantic urban scene understanding is an exciting, promising, and challenging task that is required for autonomous driving. We see that there is a large interest and progress in general scene understanding and even in the automotive domain there are production systems based on object detection for individual classes. However, when targeting autonomous driving, semantic scene understanding is not at the desired level of maturity. In fact, as discussed above we are facing a large number of different classes with different shapes and properties paired with the challenges that arise in the automotive domain, c.f. Section 1.3. In addition, we need to have a detailed scene understanding that can exceed the coarse representation via bounding boxes. Seeking for a system that is in principle capable of solving these requirements, we focus on *pixel-level semantic labeling* that we make accessible to downstream applications via a compact representation called the *Stixel world*. In doing so, the semantic information is enriched with a 3D scene understanding and with a segmentation that mitigates the lack of instance separation. While pixel-level semantic labeling is an established component for general scene understanding, c.f. Section 1.2, there are several open questions that we address in this dissertation.

First, in an automotive setup, multiple input modalities such as temporal information from the video stream or depth information via stereo vision are available, but their full potential is unclear. Second, in such a setup the scene type is always street scenes and the camera is fixed in the vehicle yielding a known pose; information that could and should be exploited. Third, in recent years immense progress in scene understanding was achieved via the availability of large-scale datasets and accompanying benchmarks, e.g. ImageNet and Microsoft COCO for general scenes or KITTI for urban scenes. However, for pixel-level scene understanding in an automotive environment, there is no such data available. Fourth, an appropriate scene representation carrying the perceived information about the environment is barely considered. We argue that such a representation needs to be compact and robust to be suitable for autonomous driving. While investigating these open items throughout the dissertation, we will follow a common recipe. We scan the literature for existing solutions, identify the open issues, and address these by combining existing methods and adding what is missing. We do so by keeping the particular challenges in an automotive environment in mind and work towards a system that delivers the semantic information of the scene robustly and compactly at high efficiency.

## 1.5 DISSERTATION OUTLINE

The remainder of the dissertation is structured as follows. We start with an overview of related work in Chapter 2 with respect to the scope of the dissertation. This discussion is conducted from a rather general and high-level point of view as we address more closely related works in the individual technical chapters. In Chapter 3, we investigate the benefits of multi-cue input data as is typically available in autonomous vehicles for pixel-level semantic labeling. We design our system based on traditional processing concepts, where we extend most stages to leverage the available input modalities. Next, we enable deep learning methods for semantic street scene understanding by introducing *Cityscapes*, a large-scale dataset of urban scenes in Chapter 4. The dataset is accompanied by a benchmark based on which we compare state-of-the-art methods for pixel- and instance-level semantic labeling. In addition, we combine existing concepts and network architectures to aim for an efficient neural network for pixel-level semantic labeling at high classification accuracies. In Chapter 5, in order to combine previous lessons learned, we reformulate the multi-layer *Stixel World* as first proposed by Pfeiffer and Franke, 2011b as a graphical model, which provides a clear mathematical formalism explicitly modeling statistical independence assumptions. This formulation in turn allows us to integrate multiple input cues, in particular depth from stereo vision and pixel-level semantic labels, yielding the *Semantic Stixel World*. The obtained representation is a segmentation tailored for street scenes, combines 3D with semantic information, and has properties desirable for autonomous vehicles, such as efficiency, compactness, and robustness. Eventually, we conclude the dissertation in Chapter 6 with a discussion and an outlook for future work.

### 1.5.1 Contributions

**EFFICIENT MULTI-CUE SEMANTIC LABELING** Applying semantic labeling to a specific scenario such as autonomous driving requires high accuracies of the system’s output obtained in real-time. Simultaneously, the scene type is restricted to street scenes and typically multiple input modalities, such as depth from stereo vision, motion from optical flow, or object detections, are available. Despite these requirements and peculiarities of autonomous driving, semantic labeling approaches are commonly general-purpose methods, are rather slow, and, if at all, they leverage only a few additional input channels for a few processing stages. In contrast, in Chapter 3, we propose a semantic labeling system tailored for the application of autonomous driving. We design our model around the given scene type, exploit that the camera pose is nearly constant with respect to the environ-

ment, and leverage available input cues holistically throughout all processing stages. In doing so, we show that these cues are complementary and help to improve the overall accuracy. Our method is centered around a segmentation tree that already exploits multiple cues to generate accurate region proposals. Its tree structure allows for efficient inference, which again exploits all available input modalities. Overall, our approach achieves excellent accuracies at real-time speeds.

**CITYSCAPES DATASET AND BENCHMARK SUITE<sup>10</sup>** The recent success of deep learning methods for various tasks in semantic scene understanding is mainly driven by the availability of large-scale datasets enabling supervised training of deep neural networks. However, prior to this work, there was no large-scale dataset and benchmark available for the task of urban semantic labeling. To this end, we proposed the Cityscapes dataset and benchmark suite that we introduce in Chapter 4, where we address large-scale urban scene understanding. The dataset consists of several thousands of images with fine and coarse annotations for training and testing methods for pixel- and instance-level semantic labeling. In addition to the images and their annotations, the dataset includes additional modalities, such as depth, video, or vehicle odometry. We provide a discussion of our major design choices, an extensive evaluation of the dataset’s statistics and compare to related works. The dataset is accompanied by a benchmark suite, based on which we evaluate and compare both general-purpose and specific deep learning methods for the task of urban scene understanding. Analyzing these methods and combining the lessons learned, we propose a competitive real-time deep model for pixel-level semantic labeling. Overall, Cityscapes became today’s benchmark for urban semantic scene understanding in our research community.

**SEMANTIC STIXELS** For autonomous driving, the scene needs to be compactly represented such that the bandwidth required to transfer the information to further downstream modules is reduced. Furthermore, the environment model needs to be robust, efficient to compute, and should contain essential information for autonomous driving. To this end, Pfeiffer and Franke, 2011b proposed the *Stixel World*, a segmentation of the scene based on depth information yielding the drivable space and delimiting obstacles. In Chapter 5, we extend this model and introduce *Semantic Stixels*. We formalize the *Stixel World* via a graphical model that allows to fuse multiple input channels supported by our findings in Chapter 3. Furthermore, we provide an analysis of the inference scheme and show how to learn the model

<sup>10</sup> The creation of the Cityscapes dataset and benchmark suite was initially driven by Marius Cordts and was then continued as joint work with Mohamed Omran.

parameters from training data. Overall, the model is capable of accurately representing the scene in terms of depth and semantic information, while the latter stems from an object detector or a real-time semantic labeling network as proposed in Chapter 4.

## RELATED WORK

## CONTENTS

2.1	Overview of Pixel-level Semantic Labeling . . .	13
2.1.1	Classic processing pipeline . . . . .	13
2.1.2	Deep learning . . . . .	15
2.2	Constrained Scene Type . . . . .	17

The goal of this chapter is to embed the dissertation into the vast body of related work. We will do so from a high-level perspective and leave the detailed analysis and differentiation of closely related work to the technical Chapters 3 to 5. As the focus of this dissertation is pixel-level semantic labeling, we also focus the analysis of related work on this task and provide an overview in Section 2.1. Subsequently, in Section 2.2, we turn towards scene understanding methods that focus on a constrained scene type and exploit the resulting domain restrictions; a concept that we also follow in this work.

## 2.1 OVERVIEW OF PIXEL-LEVEL SEMANTIC LABELING

Methods for pixel-level semantic labeling, also denoted as *scene labeling* or *semantic segmentation*, can be roughly split into two lines of research, even though exceptions and hybrid variants exist. We start with a discussion of the *classic processing pipeline* (Section 2.1.1) and then address state-of-the-art *deep learning* methods that are trained end-to-end (Section 2.1.2). This structure is also reflected in this dissertation, where we start with an efficient semantic labeling system following a classic processing pipeline in Chapter 3. We then turn towards deep learning-based methods by providing an appropriate large-scale dataset in Chapter 4 that we use to benchmark different variants in terms of accuracy and run time. Eventually, we combine both methodologies yielding our compact Stixel representation in Chapter 5.

2.1.1 *Classic processing pipeline*

From a high-level perspective, classic methods for semantic labeling are often based on a common processing pipeline that consists of bottom-up image segments, feature extraction, region classification, and final inference. Naturally, there exist many variants of this



pipeline, where one or multiple components are missing, altered, or added, and there are methods that follow widely unrelated concepts, e.g. Brox et al., 2011; L.-C. Chen et al., 2013; Fröhlich et al., 2012; Yang et al., 2010.

The first step in the processing pipeline consists of the computation of bottom-up image segments. In Fulkerson et al., 2009; Mičušík and Košecká, 2009; Tighe and Lazebnik, 2013, these segments are simple superpixels, e.g. the Graph Based Image Segmentation (GBIS) by Felzenszwalb and Huttenlocher, 2004 or quick shift (Vedaldi and Soatto, 2008). Other approaches such as Ladický et al., 2013; Plath et al., 2009; Reynolds and Murphy, 2007 also rely on such superpixels but computed at multiple scales to retrieve a set of overlapping segments. A third group of works (Arbeláez et al., 2012; Lim et al., 2009; Nowozin et al., 2010; Yao et al., 2012) leverage segmentation methods based on edge detectors, such as the gPb-OWT-UCM segmentation tree by Arbeláez et al., 2011. Researchers apply such bottom-up segmentations in their work for various reasons. First, these unsupervised regions aid the semantic segmentation since they can provide precise boundaries of the semantic classes. Second, they can serve as a regularizer, because a single segment typically has a constant semantic class. Third, the segments can serve as proposal regions for a classifier and aid the classification via an increased spatial support. Lastly, the bottom-up segmentation can increase the efficiency if inference is conducted on the superpixel- instead of the pixel-level.

As a second step, feature descriptors for the bottom-up image regions are computed in order to facilitate their classification into the semantic classes. The feature extraction step is typically split into two parts. First, one or multiple pixel-level feature representations such as textons (Malik et al., 2001), SIFT (Lowe, 2004), Gabor filters, or the color values themselves are computed, e.g. Fulkerson et al., 2009; Plath et al., 2009; Reynolds and Murphy, 2007. Second, these vectors are *pooled* over all pixels within the region in order to obtain a fixed length, compact, and discriminative descriptor of the region. Common pooling techniques include sum-, average-, or max-pooling, c.f. Boureau et al., 2010; Carreira et al., 2012a. Inspired by the Bag-of-Words (BoW) model from natural language processing (Lewis, 1998), the Bag-of-Features (BoF) model in computer vision has emerged. Here, prior to pooling, the high-dimensional and often real-valued feature vectors are encoded, i.e. mapped onto an element of a finite *codebook*, e.g. via k-Means clustering (Fulkerson et al., 2009; Ladický et al., 2013; Plath et al., 2009), or decision trees (Moosmann et al., 2008; Scharwächter et al., 2013). This encoding can be thought of replacing the pixel-level feature vectors with vectors that contain a single one at the index of the *codeword*, and zeros otherwise. If these vectors are then pooled via average-pooling, one essentially computes the normalized histogram over the codewords describing the region.



In addition to pixel-level appearance features, prior work also employs various other region descriptors, e.g. based on geometric and shape properties of the region (Arbeláez et al., 2012; Lim et al., 2009; Tighe and Lazebnik, 2013).

Third, the region descriptors are classified into the semantic classes of interest by leveraging standard classifiers. These classifiers typically yield likelihood scores for each semantic class. Often a Support Vector Machine (SVM) or a variant thereof is used, e.g. Arbeláez et al., 2012; Fulkerson et al., 2009; Plath et al., 2009, but also a nearest neighbor variant is employed by Tighe and Lazebnik, 2013, or AdaBoost, as proposed by Freund and Schapire, 1996, is used by Ladický et al., 2013; Reynolds and Murphy, 2007. Yao et al., 2012 directly classify pixels and pool the likelihoods (instead of features) over the image regions.

In the last stage, the final pixel-level labeling is inferred. In the simplest case, the region labels could be directly projected onto the pixels, however, the results are then typically noisy and unsatisfying. Instead, many works model the pixel label result via a graphical model, e.g. a Markov Random Field (MRF) in Tighe and Lazebnik, 2013, a Conditional Random Field (CRF) in Gonfaus et al., 2010; Kohli et al., 2009 or the pylon model in Lempitsky et al., 2011. Such models include the class likelihoods as unary terms and extend these with prior knowledge as well as higher-order dependencies to regularize and improve the labeling result. For an excellent overview on graphical models the reader is referred to Blake and Kohli, 2011; Nowozin and Lampert, 2011. Often the nodes in the graphical models correspond to image regions in order to capture long-range dependencies while keeping the computational complexity tractable, but sometimes there are also pixel-level nodes, e.g. Ladický et al., 2013. The parameters of the graphical model are either manually tuned, are learned from training data, or in an extreme the feature classification step is skipped and the feature vectors directly form the unary potentials of a CRF, c.f. Nowozin et al., 2010. Besides graphical models, other works also use custom inference schemes (Lim et al., 2009) or another set of classifiers based on the region scores (Arbeláez et al., 2012).

### 2.1.2 Deep learning

With the availability of large-scale datasets and sufficient computational resources, Deep Neural Networks (DNNs) are nowadays at the core of nearly all state-of-the-art semantic scene understanding methods. For an excellent overview on the field of deep learning the reader is referred to Goodfellow et al., 2016.

The popular work of Shelhamer et al., 2017 breaks with the classic methodology outlined in the previous section and produces the final pixel-level semantic labeling with a single forward pass of a Fully

Convolutional Network (FCN) trained end-to-end for pixel-level semantic labeling. Shelhamer et al., 2017 base on a network designed for image classification, i.e. VGG-16 (Simonyan and Zisserman, 2015), and transform the inner product layers to convolutional layers such that the network can be applied to images of arbitrary dimensions. In doing so, they can initialize the semantic labeling model with a network that was trained for large-scale image classification and hence has learned generic image representations (Razavian et al., 2014). Due to various pooling layers in the network, the obtained output dimensions are 32 times smaller than the input in both directions. In order to regain the original resolution, Shelhamer et al., 2017 propose *skip connections* that combine the feature maps from multiple layers (and resolutions) to obtain the final feature map, which is only 8 times smaller than the input. Hence, this model is denoted as FCN-8s. The final pixel-level labels are then computed via bilinear interpolation of the confidence scores. In this dissertation, we base on the FCN-8s architecture by Shelhamer et al., 2017 for our experiments in Chapter 4 and for providing the semantic class information in Chapter 5.

Following this line, Badrinarayanan et al., 2017; Ghiasi and Fowlkes, 2016; G. Lin et al., 2017; Wang et al., 2017 propose alternative strategies based on sophisticated Convolutional Neural Network (CNN) modules to upscale the coarse feature maps to the desired output resolution. Alternatively, L.-C. Chen et al., 2016; Yu and Koltun, 2016 leverage dilated convolutions that can circumvent the downscaling in the network altogether while maintaining its receptive field, which is important for context knowledge. Pohlen et al., 2017 design a network architecture that couples a full resolution processing stream with a standard downscaling one. Others aim to incorporate global scene context for pixel-level prediction, in particular through a dedicated network architecture (W. Liu et al., 2015; Sharma et al., 2015; Yu and Koltun, 2016; Zhao et al., 2017) or through Recurrent Neural Networks (RNNs; Byeon et al., 2015; Pinheiro and Collobert, 2014). Naturally, a fourth line of work is to improve performance by basing on better image classification networks, e.g. Z. Wu et al., 2016.

Other work aims to bridge the gap towards the classic pipeline introduced in Section 2.1.1. One option is to leverage the CNN to compute pixel-level features or semantic scores and then employ superpixels for regularization and precise segmentation (Farabet et al., 2012; Mostajabi et al., 2015; Sharma et al., 2015). Another option is to combine the strengths of CNNs and CRFs. Typically, the CRF is appended to an FCN such that it improves the output via regularization, the modeling of long-range dependencies, and the incorporation of prior knowledge. The CRF can be simply appended to the network as an individual processing block (L.-C. Chen et al., 2016) or can be incorporated in an end-to-end training (Arnab et al., 2016; G. Lin et al., 2016; Z. Liu et al., 2015; Schwing and Urtasun, 2015; S. Zheng

et al., 2015). Note that in the latter case, the CRF is essentially another layer (or multiple) at the end of the CNN. In a third variant, Xiaoxiao Li et al., 2017 combine cascaded models with deep learning. Multiple smaller networks are stacked on top of each other, while earlier stages classify easy image regions and later stages are responsible for the hard cases; nevertheless, the whole model is trained end-to-end.

Only a few works have focused on efficient semantic labeling for real-time autonomous driving applications. Trembl et al., 2016 rely on SqueezeNet (Iandola et al., 2016) as underlying image classification network, where the non-linearities are replaced by exponential linear units (Clevert et al., 2016). In contrast, Paszke et al., 2016 develop a network architecture specifically tailored for efficient pixel-level semantic labeling.

## 2.2 CONSTRAINED SCENE TYPE

In this dissertation, we focus on pixel-level semantic labeling for autonomous driving. In such a constrained scenario, a scene understanding system can exploit various constraints on the scene type, i.e. street scenes, recorded from a moving vehicle with nearly fixed camera pose with respect to the environment. In addition, meta data such as depth and motion information via stereo vision and optical flow, respectively, is readily available (Franke et al., 2013; Geiger et al., 2013). Throughout the thesis, we exploit this prior knowledge. In Chapter 3, we propose a semantic labeling system that leverages multiple input cues throughout its processing stages. In Chapter 4, we introduce a large-scale dataset that is recorded in urban scenes and is accompanied with a benchmark that we use to compare generic semantic labeling methods for the specific scene type at hand. Eventually, we combine what we have learned and propose a compact semantic scene representation that is tailored for urban scenes. In order to embed this dissertation in the literature, we therefore turn our analysis of related work in this section towards those methods that explicitly exploit such domain knowledge. We also include methods for other scene types than urban street scenes as well as for other scene understanding applications than semantic labeling in our discussion. Our analysis in this section can be grouped into two major categories.

In our first category, we address methods that leverage constraints on the overall scene layout. Typical outdoor scenes share a common layered structure of a ground surface at the bottom, the sky at the top, and objects in between, which is modeled by Felzenszwalb and Veksler, 2010; M.-Y. Liu et al., 2015; Scharwächter et al., 2013; Y. Zheng et al., 2012, whereas A. Gupta et al., 2010 model the environment via 3D geometric blocks. Focusing on the dynamic objects in street scenes, i.e. mainly vehicles and pedestrians, prior work exploits that these objects are typically located on a common ground plane at various

distances. This knowledge can be used to predominantly improve accuracy (X. Chen et al., 2015; Pan and Kanade, 2013) or run time (Enzweiler et al., 2012; Leibe et al., 2007) as well as for explicit reasoning about occlusion (Wojek et al., 2013) and scene layout (Geiger et al., 2014). Riemenschneider et al., 2014; Valentin et al., 2013 transform the scene at hand into 3D meshes and base their method on this altered representation. Indoor scenes typically impose less overall constraints, but nevertheless exhibit a common layout that is considered by the models of D. Lin et al., 2013; Silberman et al., 2012; J. Zhang et al., 2013.

Our second group discusses multi-cue approaches. When facing a constrained scenario, there are often additional cues available, such as depth estimation via stereo vision and laser scanners for street scenes (Geiger et al., 2013), or structured light and time of flight cameras for indoor rooms (Song et al., 2015). Pixel-level semantic labeling performance is improved using features based on the depth information of indoor scenes by Couprie et al., 2013; Deng et al., 2015; S. Gupta et al., 2015, 2014, 2016; Hazirbas et al., 2016; Khan et al., 2016; Z. Li et al., 2016; Müller and Behnke, 2014; X. Ren et al., 2012. Furthermore, Eitel et al., 2015; Hou et al., 2016 leverage depth cues for indoor object detection and Silberman et al., 2012 for support structure inference. Kreso et al., 2016; C. Zhang et al., 2010 use stereo vision, Miksik et al., 2013 optical flow, Scharwächter et al., 2014b both, and R. Zhang et al., 2015 laser scans to aid the semantic labeling of street scenes. Kundu et al., 2014; Sturgess et al., 2009 exploit that a vehicle is driving through a widely static 3D world, which is inferred together with the semantic labeling. The same concept additionally supported by stereo vision is at the core of the approaches by Floros and Leibe, 2012; H. He and Upcroft, 2013; Sengupta et al., 2013. Ardeshir et al., 2015 leverage a geographical information system that provides additional information about urban areas as well as GPS for pixel-level semantic labeling. Eigen and Fergus, 2015; Ladický et al., 2014 leverage depth cues during training such that the model learns about the typical 3D scene structure and can infer the structure at test time jointly with the pixel labeling.

## CONTENTS

3.1	Introduction . . . . .	20
3.2	Encode-and-Classify Trees . . . . .	22
3.2.1	Pixel-level semantic labeling . . . . .	24
3.2.2	Texton maps . . . . .	25
3.2.3	Combination . . . . .	26
3.2.4	Training . . . . .	26
3.3	Superpixels . . . . .	27
3.3.1	Incorporating depth . . . . .	28
3.3.2	Incorporating pixel classification . . . . .	28
3.4	Multi-cue Segmentation Tree . . . . .	29
3.4.1	Proximity cues . . . . .	30
3.4.2	Tree construction . . . . .	34
3.5	Multi-cue Region Classification . . . . .	35
3.5.1	Multi-cue features . . . . .	35
3.5.2	Training . . . . .	37
3.6	CRF Inference . . . . .	38
3.6.1	Unaries . . . . .	39
3.6.2	Parent-child . . . . .	39
3.6.3	Inference . . . . .	40
3.7	Temporal Regularization . . . . .	40
3.8	Experiments . . . . .	41
3.8.1	Cues . . . . .	41
3.8.2	Datasets . . . . .	42
3.8.3	Metrics . . . . .	42
3.8.4	Comparison to baselines . . . . .	45
3.8.5	Ablation studies . . . . .	49
3.8.6	Qualitative results . . . . .	50
3.8.7	Run time . . . . .	50
3.9	Discussion . . . . .	53

In this chapter, we investigate the benefits of leveraging multiple complementary cues for pixel-level semantic labeling. To this end, we propose a multi-cue segmentation tree that yields accurate region proposals. Based on these proposals, we infer the semantic labeling, again with the help of multiple cues. Experiments show that competitive results on two small-scale urban scene understanding datasets are obtained at real-time speeds. The chapter is based on Cordts et al., 2017a and contains verbatim quotes of that work. The article was

the result of joint work with Timo Rehfeld. His contributions are the encode-and-classify trees, which provide the appearance information in our multi-cue system and which we hence present as related work in Section 3.2. His implementation and training methodologies of random forests also form the basis of additional classifiers that we propose in the following.

We start this chapter with an introduction in Section 3.1 and the presentation of encode-and-classify trees in Section 3.2. Subsequently, we describe the individual components of our processing pipeline, consisting of superpixels (Section 3.3), a multi-cue segmentation tree (Section 3.4), the region classification step (Section 3.5), CRF inference (Section 3.6), and a temporal filtering (Section 3.7). Eventually, we evaluate the proposed concept (Section 3.8) and discuss the observations (Section 3.9).

### 3.1 INTRODUCTION

The setting of a general scene understanding can be best analyzed by inspecting the corresponding datasets, most prominently *PASCAL VOC* (Everingham et al., 2014), *ILSVRC* (Russakovsky et al., 2015) and *Microsoft COCO* (T.-Y. Lin et al., 2014). These datasets have in common that they contain images from *the wild*, i.e. as found on image hosting websites such as Flickr, 2016, c.f. Chapter 1. Therefore, the general scene understanding task requires a large number of object classes to be recognized and allows only to impose few constraints on the overall scene layout.

In contrast, this dissertation focuses on task-specific settings that typically involve a more restricted domain, such as indoor scenes in *NYUD<sub>2</sub>* (Silberman et al., 2012) or outdoor street scenes in *KITTI* (Geiger et al., 2013) and *DUS* (Scharwächter et al., 2013). In those scenarios, the number of different object classes to be recognized is typically much smaller; moreover, assumptions on the scene layout aid the recognition task. While simplifying the problem somewhat, a number of significant challenges remain, such as highly varying object scale and motion, partial occlusions, and strong demands on computational efficiency, e.g. for real-time mobile or robotics applications. Hence, a number of approaches have focused on these task-specific domains (S. Gupta et al., 2014; Scharwächter et al., 2014a; Sengupta et al., 2013; Yao et al., 2012).

In this chapter, we address the task of urban semantic labeling by introducing a system that combines, integrates, and extends well-known components in an efficient way in order to yield excellent results with lowest possible execution time. The proposed approach follows a common and well established processing pipeline, c.f. Figure 3.1 and Section 2.1.1. We start by computing a superpixel segmentation of the input image. A superpixel is considered as the smallest

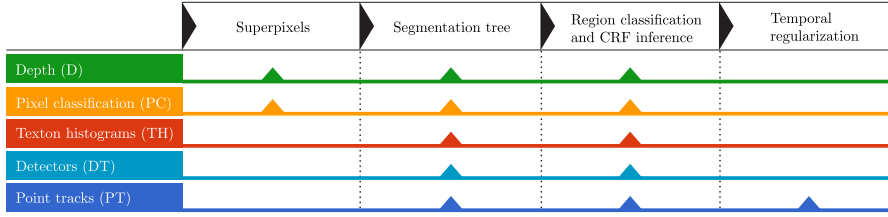


Figure 3.1: System overview. Arrows indicate the contribution of each available cue (rows) to the individual processing steps (columns).

granularity of the image, which means that all pixels within the same superpixel get assigned to the same semantic class label. Based on such an over-segmentation, region proposals are generated in form of a segmentation tree. These proposals are then classified into semantic classes and label consistency is ensured by a Conditional Random Field (CRF). Eventually, we apply a temporal regularization step. We leverage Randomized Decision Forests (RDFs) as classifiers throughout various processing stages.

When comparing methods with a similar system architecture, we make three basic observations that lead to the research goals of this chapter. First, cues that are complementary to the image channel are readily available (Franke et al., 2013; Geiger et al., 2013) and have been shown to significantly improve recognition performance, e.g. depth cues from stereo or RGB-D (Floros and Leibe, 2012; S. Gupta et al., 2014; Kähler and Reid, 2013; Scharwächter et al., 2014a), object detectors (Arbeláez et al., 2012; S. Gupta et al., 2014; Yao et al., 2012) or motion (Floros and Leibe, 2012; Kähler and Reid, 2013; Mičušík and Košecká, 2009; Scharwächter et al., 2014a). However, such cues have not been exploited to their full extent for the task of semantic labeling, since typically only a few cues are applied to some of the processing stages. Therefore, in the remainder of this chapter we will investigate the benefits of such cues throughout the various processing steps of our system. In particular we leverage appearance cues and depth estimations from stereo vision as well as temporal information via sparse point tracks and bounding box object detections (if available).

The second observation deals with the fact that high quality region proposals are crucial for classification. A multitude of approaches, e.g. Arbeláez et al., 2012; Farabet et al., 2012; S. Gupta et al., 2014; Lempitsky et al., 2011; Lim et al., 2009; Nowozin et al., 2010; X. Ren et al., 2012; Silberman et al., 2012; Yao et al., 2012, relies on the general-purpose hierarchical segmentation of Arbeláez et al., 2011, whereas others (Ladický et al., 2013; Plath et al., 2009; Reynolds and Murphy, 2007) construct a segmentation tree from multiple runs of standard superpixel algorithms, e.g. GBIS by Felzenszwalb and Huttenlocher, 2004. Only very few methods are based on proposals that are specialized for the respective domain or exploit additional cues during



superpixel generation, particularly depth (S. Gupta et al., 2014; Scharwächter et al., 2014a; Silberman et al., 2012). In this work, we propose a method to compute region proposals based on all available cues and compare general-purpose ones with specialized variants.

Third, tree-structured models allow for efficient inference, c.f. Blake and Kohli, 2011; Nowozin and Lampert, 2011. We exploit this observation and center our method around a segmentation tree that allows for fast construction, feature generation, classification, and CRF inference.

With these research goals in mind, three approaches are highly related to ours. The work of Silberman et al., 2012 segments indoor scenes into support relations using RGB-D cues. General purpose superpixels (Arbeláez et al., 2011) are combined to a region proposal tree by incorporating the depth cue similar to our work. Naturally, geometric cues such as surface normals form the driving force for support inference, while we focus on semantic labeling, where depth cues can only aid the overall inference. The work of S. Gupta et al., 2014 exploits tree structures as well as multiple cues, and shows impressive results on the NYUD2 dataset (Silberman et al., 2012). To obtain region proposals, a segmentation tree is generated using a boundary detector based on appearance and depth. The resulting regions are classified using a depth-augmented CNN-based detector. Eventually, region proposals and detections are used for semantic labeling. In contrast to both of these works, we build the segmentation tree using robust and efficient features, leverage the tree structure for efficient inference, and integrate additional cues such as motion and detectors. In doing so, we propose a near real-time method with a focus on outdoor street scenes in contrast to the indoor setup in Silberman et al., 2012 and S. Gupta et al., 2014.

Also related is the efficient urban semantic labeling approach of Scharwächter et al., 2014a, which uses multi-cue semantic labeling and RDFs. Competitive results are shown by combining fast features with spatio-temporal regularization. However, they only rely on a single layer of greedily generated region proposals and thus cannot recover from errors on this level. In addition, they do not use object detectors and strongly depend on the particular choice of superpixels. We report results in comparison to their work, using the same public benchmark dataset.

### 3.2 ENCODE-AND-CLASSIFY TREES<sup>1</sup>

In order to obtain appearance information as part of our multi-cue input data, c.f. Figure 3.1, we opt for encode-and-classify trees as proposed in Cordts et al., 2017a. These classifiers are a variant of

<sup>1</sup> This section is based on the work and the contributions of Timo Rehfeld in (Cordts et al., 2017a). The section is included in this dissertation for completeness.



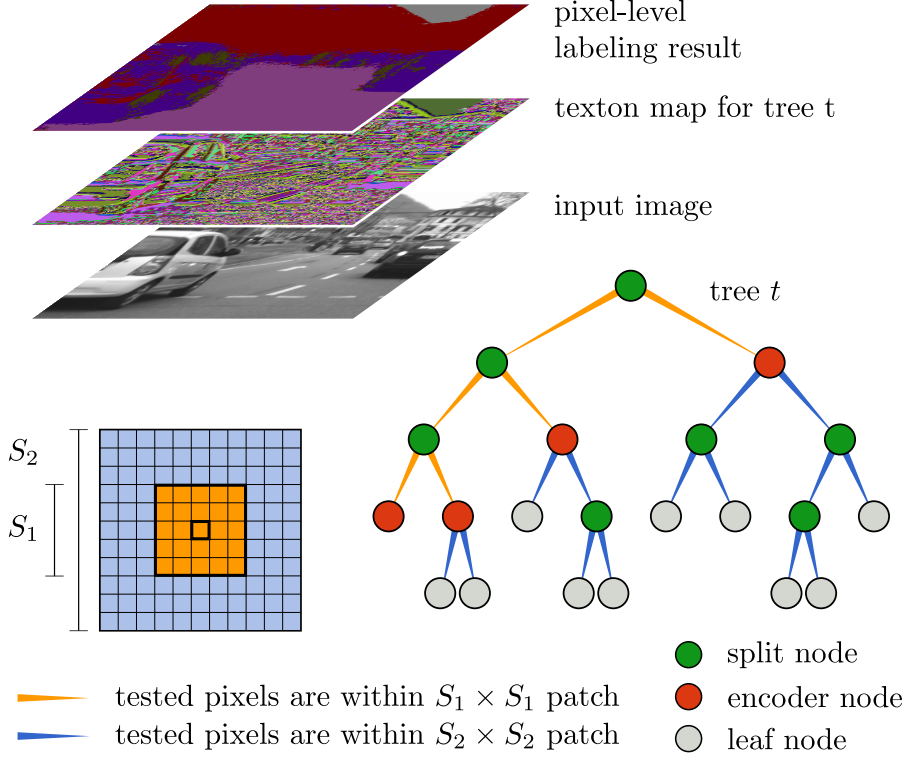


Figure 3.2: Structure of the encode-and-classify trees. The trees are used for pixel-level classification and generation of texton maps; an exemplary texton map using a single tree and the resulting pixel-level classification are shown in the top. Encode-and-classify trees are decision trees, where certain nodes are marked as *encoder nodes* (bottom right). These nodes are used to extract the texton maps and form the leaves of a sub-tree. This sub-tree operates on features extracted from rather small patches of size  $S_1 \times S_1$  (bottom left) to obtain more discriminative texton maps. The remaining nodes in the encode-and-classify trees have access to a larger patch of size  $S_2 \times S_2$  in order to achieve accurate pixel-level classification. Courtesy of Timo Rehfeld.

RDFs and simultaneously generate pixel-level semantic label scores as well as texton maps which in turn are used for segment-level bag-of-features histograms (Moosmann et al., 2008; Shotton et al., 2008). This concept is visualized in Figure 3.2. We start by discussing both tasks individually in Sections 3.2.1 and 3.2.2 and describe their combination in Section 3.2.3. We conclude the section with a description of the training procedure. Note that we introduce RDFs and their training procedure in greater detail, since we will adopt the same concepts multiple times throughout the remainder of the chapter. The output of the encode-and-classify trees delivers powerful appearance and texture information and is subsequently used for superpixel generation, segmentation tree construction, and region classification. Note that the latter two steps have no direct access to the input image and all their appearance cues stem from the encode-and-classify trees.

### 3.2.1 Pixel-level semantic labeling

The first component of the encode-and-classify trees is pixel-level semantic labeling, where the task is to assign a probability score for each considered semantic class to each pixel in the image. These probability maps can then be used as soft semantic information in the subsequent processing stages, or we can compute the maximizing class label at each pixel. These labels could in principle serve as the final output of our system, but are typically too noisy to achieve high-quality results. The pixel-level labeling consists of two steps. First, various feature channels are extracted from the input data. Second, a pixel in the image is classified using an [RDF](#) classifier based on the features within a small surrounding patch. Note that for efficiency reasons only a sub-sampled set of pixels is actually classified and the remaining pixels obtain their results via nearest neighbor interpolation. The isolated pixel labeling component has been previously proposed in Scharwächter and Franke, [2015](#).

**FEATURE EXTRACTION** The first step for the pixel-level labeling is to extract low-level feature maps of the input data. The simplest feature channel is the input image itself in the form of an illumination channel and the two normalized color components. Further, the input image is filtered using a subset of the filter-bank proposed by Leung and Malik, [2001](#). In addition, depth information from stereo vision is leveraged, which supports the discrimination of geometric classes, such as *ground* vs. *object*. The disparity map yields two feature channels: the vertical disparity gradient and the height of each pixel above a planar ground plane. For details on these feature channels, the reader is referred to Scharwächter and Franke, [2015](#).

**PIXEL CLASSIFICATION** Once the feature channels are computed, a pixel is classified using a feature vector, which is the concatenation of all feature channels within a surrounding patch of size  $S \times S$ . As classifier, an [RDF](#) is selected for pixel-level labeling, which was, prior to the recent popularization of [CNNs](#) (Chapter 4), a common choice, e.g. Fröhlich et al., [2012](#); Müller and Behnke, [2014](#); Shotton et al., [2008](#). [RDFs](#) have a fast inference time and work well on diverse features from different domains with heterogeneous scales and properties. Such an [RDF](#) is a set of trees that individually assign class probability scores to the feature vector. The final classification result is then the average over all trees. A single tree is a collection of decision stumps that are arranged in a tree structure, where each decision stump decides based on an internal criterion on the feature values whether to continue with the left or right child, c.f. Figure 3.2. Each leaf node in the tree is assigned to a class probability map that is used as the tree's output, once the leaf is reached. The decision stumps can be

arbitrary binary classifiers, however, in this work restricted to simple binary comparisons between a feature value and an internal threshold. The index of the feature value and the threshold are determined during training. For details, the reader is referred to Scharwächter and Franke, 2015.

Important for the pixel classification performance is the patch size  $S$  and a sufficient tree depth. Experiments in Cordts et al., 2017a show that medium to large patch sizes are best, since only then the required context knowledge for an accurate classification is available. Further, the tree needs to be sufficiently deep, to be able to perform discriminative decisions.

### 3.2.2 *Texton maps*

The second purpose of the encode-and-classify trees is to produce texton maps that in turn are used to generate region-level bag-of-features histograms that are used as discriminative features for region classification. Texton maps are a form a feature encoding, where a large set of feature values is condensed into a code-book of a smaller discrete set of values that facilitate discrimination. Such a feature encoding can be conducted via *RDFs*, which are in this context also referred to as Extremely Randomized Clustering (*ERC*) forests (Moosmann et al., 2008; Shotton et al., 2008). The core idea is to leverage an *RDF* for pixel classification, c.f. Section 3.2.1, but instead of assigning the leaf node probability map to a pixel, all leaf nodes in the forest are enumerated and the indices of the reached leaf nodes (one for each tree in the forest) are assigned to the pixel (Scharwächter et al., 2013). An example texton map for a single tree is shown in Figure 3.2, where indices are visualized by false-colors.

Once the texton maps are computed, they can be used to generate region-level bag-of-features histograms. A given region in the image is described by a histogram over the leaf indices assigned to the pixels within that region. These histograms form a feature descriptor of the region that can be classified into a semantic class. Due to the superior performance on these types of features, this final classification step is performed via an *SVM* with Histogram Intersection Kernel (*HIK*; J. Wu, 2010). The classifier is trained in a one-vs.-all mode and a sigmoid mapping (Platt, 1999) to convert the *SVM* output to a positive score is performed.

In contrast to the pixel classification task, it turns out that the region classification performance is best when the patch size  $S$  of the *RDF* is fairly small, c.f. Cordts et al., 2017a. Intuitively, smaller patch sizes lead to less correlation in the leaf node indices of neighboring pixels and hence the pooled histograms are more discriminative. Furthermore, the trees for feature encoding are requested to be fairly

small, since region classification run time increases quickly with increased tree depth.

### 3.2.3 Combination

The previous two sections introduced the two tasks that the [RDFs](#) have to accomplish. Since both tasks have conflicting requirements on the patch size  $S$  and the tree depth, they cannot be simultaneously addressed by standard [RDFs](#). To overcome these limitations, while keeping the computational efficiency high, encode-and-classify trees were introduced in Cordts et al., [2017a](#). The idea is to leverage sub-trees that are fairly short and have access to a small patch size  $S_1$  for feature encoding and to continue with deeper tree nodes that use features from a larger patch  $S_2$ , c.f. Figure [3.2](#). In doing so, both tasks achieve a performance close to their individual optima, while keeping the run time low.

### 3.2.4 Training

Training the encode-and-classify trees is performed in two steps and follows the work of Geurts et al., [2006](#) to create extremely randomized binary trees. First, the decision stumps are restricted to unary pixel tests based on a small patch of size  $S_1 \times S_1$  around the current pixel. To recap, this patch is represented by a feature vector, which is the concatenation of all values within the patch from various feature channels. To generate the split tests, a dimension of the feature vector or, equivalently, pixel location and feature channel, is sampled and the feature value is compared against a randomly drawn threshold. Each such split then separates the training samples  $\mathcal{D}_n$  at node  $n$  into disjoint subsets  $\mathcal{D}_l$  and  $\mathcal{D}_r$ . For each decision stump, a fixed number of split tests is sampled and the split with largest information gain

$$I = E(\mathcal{D}_n) - \frac{|\mathcal{D}_l|}{|\mathcal{D}_n|} E(\mathcal{D}_l) - \frac{|\mathcal{D}_r|}{|\mathcal{D}_n|} E(\mathcal{D}_r) \quad (3.1)$$

w.r.t. the class label distribution is chosen. The function  $E(\mathcal{D})$  denotes the Shannon entropy of the empirical probability distribution over the samples in  $\mathcal{D}$ . Training continues recursively until no further subdivision is possible, which means that all trees are trained to full depth. In other words, if all feature vectors had unique values, each leaf node in each tree is reached by a single feature vector of the training set. Since such trees are heavily over-fitted to the training set, they are pruned back bottom up until the desired number of encoder nodes is reached, giving full control over the targeted histogram length. Pruning is an iterative procedure, where nodes are randomly selected out

of those split nodes where both children are leafs. These children are then removed and the selected node becomes a new leaf and the next iteration is started. The described steps are repeated multiple times, once for each tree in the [RDF](#), while each tree is trained on a random subset of the training set, in order to increase the diversity amongst different trees. Eventually, this training procedure yields the subtrees with their leafs being the encoder nodes.

Second, starting at the encoder nodes, training is continued to obtain a well performing pixel classifier, while this time, the unary pixel tests have access to the larger  $S_2 \times S_2$  patches. Again, the [RDFs](#) are trained to full depth and random pruning is performed until a desired number of leaf nodes is obtained. During pruning, encoder nodes are excluded from becoming pruning candidates.

### 3.3 SUPERPIXELS

Superpixels are often considered as the smallest unit for semantic labeling, which means that all pixels within a superpixel are assigned to the same semantic label. Such a hard constraint can be exploited to decrease the run time of the algorithm, since many components need to reason on only a few hundred superpixels instead of millions of pixels. Further, superpixels allow to extract rich features within their image region and long-range dependencies between pixels can be efficiently modeled.

In this work, we opt for three types of superpixels: Superpixels in an Energy Optimization Framework ([SEOF](#); Veksler et al., 2010), Graph Based Image Segmentation ([GBIS](#); Felzenszwalb and Huttenlocher, 2004), and Stixels (Pfeiffer and Franke, 2011b). This selection is based on the main properties of these superpixel methods. All three approaches have in common that they either explicitly formulate weights between two adjacent pixels ([GBIS](#)) or solve an underlying energy minimization problem (Stixels), or both ([SEOF](#)). Both cases allow for a straight-forward incorporation of additional cues as described in Sections 3.3.1 and 3.3.2. Further, the three methods can be seen as representatives of different fundamental types of superpixels. In [SEOF](#) and many other superpixel methods, e.g. Simple Linear Iterative Clustering ([SLIC](#), Achanta et al., 2012), the superpixels are hard constrained to a maximum size, which enforces compactness and increases computational efficiency. In contrast, [GBIS](#) only supports compactness, but, in principle, superpixels can become arbitrarily large. While [SEOF](#) and [GBIS](#) were designed for a segmentation of generic images, Stixels are highly optimized for street scenes and model their typical geometric layout of a ground plane with perpendicular obstacles. Note that Stixels are presented in greater detail in Chapter 5.

Overall, the selection of superpixels allows to compare the influence of the different properties on the overall system performance.

Further, we demonstrate the generality of our system with respect to different superpixel methods. To bring all three variants to a comparable level, we introduce modifications as described in the following sections.

### 3.3.1 Incorporating depth

As Stixels use depth information, we also add depth to [SEOF](#) and [GBIS](#). Both superpixel variants explicitly formulate weights between two adjacent pixels. We extend these weights to use the disparity image  $\mathcal{D}$  in addition to the color or gray value image  $\mathcal{J}$  for obtaining a segmentation that accurately captures both kinds of edges.

For [GBIS](#), we define the weight between two adjacent pixels  $(p, q) \in \mathcal{N}$  as

$$w_{pq} = \frac{1}{\mu_i} |\mathcal{J}(p) - \mathcal{J}(q)| + \frac{1}{\mu_d} |\mathcal{D}(p) - \mathcal{D}(q)| . \quad (3.2)$$

The normalization terms

$$\mu_i = \frac{1}{|\mathcal{N}|} \sum_{(m,n) \in \mathcal{N}} |\mathcal{J}(m) - \mathcal{J}(n)| \quad (3.3)$$

and analogously for  $\mu_d$  are computed as the average absolute differences of all adjacent pixels and balance both channels against each other.

For [SEOF](#) the weights are defined similarly as

$$w_{pq} = e^{-\frac{(\mathcal{J}(p) - \mathcal{J}(q))^2}{2\sigma_i^2}} + e^{-\frac{(\mathcal{D}(p) - \mathcal{D}(q))^2}{2\sigma_d^2}} . \quad (3.4)$$

Again, the normalization is the average contribution of all neighborhoods and for  $\sigma_i$  computed as

$$\sigma_i^2 = \frac{1}{|\mathcal{N}|} \sum_{(m,n) \in \mathcal{N}} (\mathcal{J}(m) - \mathcal{J}(n))^2 . \quad (3.5)$$

Finally, the median disparity is assigned to each superpixel in order to obtain a robust depth estimate. Superpixels without valid disparity measurements due to stereo occlusions are labeled as *void* and removed from further processing, c.f. the transparent areas in [Figure 3.3](#).

### 3.3.2 Incorporating pixel classification

Stixel superpixels are extracted from depth information only and do not take the gray value or color information into account. To also make Stixels comparable to [GBIS](#) and [SEOF](#), we alter the computation to use the pixel classification scores as an additional data term, as





Figure 3.3: Three superpixel variants we employ at the lowest level of our region proposal tree. From left to right: [SEOF](#) (Veksler et al., 2010), [GBIS](#) (Felzenszwalb and Huttenlocher, 2004), and [Stixels](#) (Pfeiffer and Franke, 2011b). In all variants, superpixels on the ground surface and sky region are already grouped (see text) and not considered in the segmentation tree. Superpixels with invalid depth information are ignored and visualized transparently.

presented in Scharwächter and Franke, 2015. These modifications help especially in regions with weak disparity information, e.g. the sky.

Since the Stixel representation explicitly separates ground and sky regions, we also do so for [GBIS](#) and [SEOF](#). Specifically, we first compute the average score of our pixel classifier in each superpixel. Superpixels for which *sky* or *ground* have maximal average classification scores are removed from further consideration. Consequently, all three variants deliver a comparable representation, visualized in Figure 3.3, where the remaining *obstacle* superpixels are highlighted in red. The segmentation tree as introduced in Section 3.4 is constructed from these *obstacle* superpixels only.

### 3.4 MULTI-CUE SEGMENTATION TREE

A central component in our processing pipeline, is the multi-cue segmentation tree, which provides region proposals for later processing. While superpixels typically provide a non-overlapping over-segmentation of the image such that no meaningful edges are missed, region proposals are requested to be larger and represent hypotheses of image regions with a single semantic class. Such a proposal set typically contains overlapping regions forming an over-complete set such that ideally each object in the scene is accurately covered by one region. This maximizes its spatial support, which is especially beneficial for the robustness of region classifiers as used in Section 3.5, since reasoning about the geometry of an object is supported by an accurate segmentation of the object as a whole. Further, appearance-based classifiers perform better on larger regions and therefore a maximal spatial support is desired. Overall, the task of the segmentation tree

is to provide overlapping region proposals, such that the objects in the scene are covered by accurate segments, while at the same time keeping the number of regions low for computational efficiency.

We opt for region proposals in form of a segmentation tree that is constructed by iteratively merging image regions, starting with superpixels, c.f. Section 3.3, as the leaf nodes. Such a tree structure has two main advantages. First, street scenes have a significant range of scales, meaning that there are often simultaneously objects close to the camera and occupying large image regions, as well as distant or occluded objects with only a few pixels of size. Such a variation of scale matches well with a segmentation tree, where the lower levels provide small proposals, while segments in higher levels span large image regions. Second, the tree structure allows for a computationally efficient system design. Features on a region level, such as histograms, geometric information, occlusion cues, and average pixel classifier scores, can be computed in a cumulative fashion. Due to the tree structure, they need to be computed only once for all superpixels on the lowest level, and can then be accumulated from leaf to root.

Because the segmentation tree consists of multiple levels, each superpixel is covered by several overlapping regions. The final label decision is thus made in a subsequent inference stage; a tree-structured CRF based on the regions in the segmentation tree allows to do this in an efficient and globally consistent way.

#### 3.4.1 Proximity cues

For constructing the segmentation tree, we start by grouping superpixels based on similarity measures. Instead of hand-crafting these measures, we use a binary classifier on adjacent superpixels incorporating multiple proximity cues. The definitions of these features are summarized in Table 3.1, while we discuss the motivation behind the individual cues in the following. All features are symmetric and are designed to be invariant across scale.

**2D REGION GEOMETRY** Solely from visually inspecting the 2D segments, one gets a rough estimate about which superpixels are more or less likely to belong to the same object. This human feeling is mainly driven by the relative location of the superpixels. Therefore, we design a set of features that is capable to compactly describe such properties. Four features ( $G_1$ – $G_4$ ) are based on the superpixel’s bounding boxes and are computed as the distances of the four respective edges, e.g. the distance in pixels between the top edges of the two bounding boxes of adjacent superpixels. The next three features ( $G_5$ – $G_7$ ) describe the relative location of the two region centers, i.e. horizontal and vertical distance, as well as the absolute sine of the angle between their connecting line with the horizontal axis. Feature  $G_8$  captures the



---

*2D region geometry*

G1, G2	bounding boxes: abs. top (bottom) coordinate difference
G3, G4	bounding boxes: abs. left (right) coordinate difference
G5, G6	absolute center row (column) difference
G7	horizontalness, i.e. absolute sine of the angle $\alpha$ between the line connecting the region centers and the horizontal axis: $ \sin \alpha  = G_5 / \sqrt{G_5^2 + G_6^2}$
G8	boundary pixel (BP) ratio: $ \cap BP  /  \cup BP \setminus \cap BP $
G9	relative size (S) difference: $ S_1 - S_2  / S_1 + S_2$

*Depth*

D1, D2	absolute depth (disparity) difference
D3, D4	average depth (disparity)
D5	D1 minus depth uncertainty at average depth
D6-D9	3D absolute top (right, bottom, left) difference
D10	average center height above ground

*Pixel classification*

PC1	inner product of average pixel classifier scores
PC2	agreement of arg max pixel classifier labels: 0 if same label, 2 if different label, value shifts towards 1 with decreasing confidence

*Texon histograms*

TH1	texon histogram intersection
-----	------------------------------

*Detectors*

DT1	joint coverage by vehicle detector: 2 if both, 1 if one, 0 if none of both superpixels covered by bounding box
DT2	DT1 with pedestrian detector

*Point tracks*

PT1	cosine of angle between velocity vectors
PT2	absolute velocity difference [ $\text{m s}^{-1}$ ]
PT3	average velocity [ $\text{m s}^{-1}$ ]

---

Table 3.1: Features for segmentation tree construction. Each feature is extracted for a pair of adjacent superpixels. A binary [RDF](#) classifier is trained on these features to decide whether the pair of adjacent superpixels should be grouped or not.

proximity in terms of a common boundary and is defined as the ratio of common boundary pixels to disjoint ones. The last 2D geometric feature  $G_9$  reflects the region sizes and is computed as the ratio of size difference (in pixels) to the sum of both sizes.

**DEPTH** Since superpixels belonging to the ground plane or the sky are removed from further processing based on the pixel classification, c.f. Section 3.3.2, the segmentation tree is constructed for the remaining objects only. In street scenes, such objects are typically perpendicular to the ground plane and are located at different distances. Thus, the gap in depth between two neighbored superpixels ( $D_1$ ) is a strong cue for region proposal generation, c.f. Scharwächter et al., 2014a. However, since the noise of depth measurements as obtained by stereo vision increases quadratically with the distance, we do not rely on this feature alone. Instead, we add the absolute disparity difference ( $D_2$ ), as well as the average depth and disparity values of the two superpixels ( $D_3$ ,  $D_4$ ). In doing so, the proximity classifier can learn to trust a possible depth gap differently, depending on the superpixel’s distances. To even further support such a reasoning, we add  $D_5$ , which is the absolute distance difference reduced by the uncertainty of distance measurements at the superpixels’ average distance, according to J. Schneider et al., 2016.

Another set of features is based on the 3D bounding boxes around superpixels. Since depth measurements from stereo can be noisy and are prone to outliers, such a bounding box needs to be computed robustly. In order to do so, we leverage the median disparity that is assigned to each superpixel, c.f. Section 3.3.1, and project the 2D bounding box into the 3D space, yielding robust superpixel extents in horizontal and vertical direction, but not in the direction of the camera axis. The differences between the four respective edges of these bounding boxes yields  $D_6$ – $D_9$ . Eventually, we add  $D_{10}$ , which is the average height above ground of the two superpixels to allow for a different reasoning at various height levels in the scene.

**APPEARANCE** Our appearance-based similarity measures are provided by the encode-and-classify trees: pixel classifier label agreement ( $PC_1$ ,  $PC_2$ ) and texton histogram intersection ( $TH_1$ ). The intuition behind the pixel classifier features is that two superpixels belonging to the same object should agree in terms of the semantic labels of the covered pixels as predicted by the pixel classification, c.f. Section 3.2.1. Therefore, the pixel-level semantic scores are averaged over all pixels within each superpixel, yielding a more robust score vector.  $PC_1$  is the inner product of these vectors of two adjacent superpixels and expresses the label agreement while considering all semantic scores. In addition, we determine the most likely labels  $l_a$

and  $l_b$  as well as their probability scores  $p_a$  and  $p_b$ . Then the feature PC2 is defined as

$$PC2 = \begin{cases} 1 - \min(p_a, p_b) & \text{if } l_a \neq l_b \\ 1 + \min(p_a, p_b) & \text{otherwise} \end{cases} \quad (3.6)$$

This value is between zero and one, if both labels disagree, and between one and two otherwise. If both labels have a high confidence the value is close to zero upon disagreement and close to two upon agreement. In case of at least one small confidence score the value is close to one.

In addition to the label agreement, we encode the appearance similarity by leveraging the texton histograms obtained by the encode-and-classify trees, c.f. Section 3.2.2. Inspired by the Histogram Intersection Kernel (HIK, J. Wu, 2010) as used in SVM classifiers to compare histograms, we use the texton histogram intersection between two superpixels as feature TH1. These histograms are designed to encode the appearance of a region and hence their distance is a powerful proximity score.

**DETECTORS** Often, additional semantic knowledge from bounding box detectors, e.g. for Advanced Driver Assistance Systems (ADAS), is readily available. Such bounding boxes typically contain whole objects and therefore provide strong proximity cues for our superpixels. For each available detector (*vehicle* and *pedestrian* in our experiments, c.f. Section 3.8.1), we add a similarity feature DTi that can have three discrete values: 2 if two adjacent superpixels are predominantly covered by an identical bounding box, 1 if not, but at least one superpixel is covered by a bounding box, and 0 otherwise.

**POINT TRACKS** The last cue that we consider in this work, are sparse point tracks. These are typically obtained via local feature trackers that match feature points in temporarily consecutive frames. Paired with vehicle odometry and stereo vision, we obtain 3D position and velocity estimations for each track, please refer to Section 3.8.1 for our experimental setup. To obtain a robust velocity estimate of each superpixel, we compute the median velocity vector of all tracks within each superpixel for each principal axis.

Since most objects in street scenes are either static or move rigidly, we expect superpixels that belong to the same object to move in the same direction with the same speed. Thus, we compute the features PT1 and PT2 being the absolute speed and direction difference (cosine of angle between velocity vectors), respectively. Since the measurement noise increases with increased speed, we follow a similar strategy as with the depth cues and the stereo noise, see above. We add the feature PT3, which is the average velocity of both superpixels, such that the proximity classifier can trust these features differently, depending on the absolute speed.

### 3.4.2 Tree construction

Using the proximity features as introduced in the previous section, we train a binary classifier to group or separate pairs of superpixels. This task is related to the boundary strength classifier in Silberman et al., 2012, the same-label classifier in Hoiem et al., 2007, or classifier-based pairwise potentials in CRFs, e.g. Huang et al., 2011. As classifier, we opt for a Randomized Decision Forest (RDF), since these classifiers can cope well with such a diverse set of features. For training we follow the strategy described in Section 3.2.4, i.e. we train until full depth and apply random pruning. We treat all adjacent superpixels with identical majority ground truth label as positive samples and other pairs as negative. If both superpixels have the *void* label assigned, they do not contribute as training sample.

After running the trained classifier on all pairs of adjacent superpixels, we assign the classifier scores as weights to the respective connecting edges. The segmentation tree is constructed using Algorithm 3.1, which only requires two parameters, the merging rate  $p$  and the number of levels  $n_l$ . Both parameters are independent of the superpixel method or the actual number of superpixels. During the edge update in the last step of the while loop in Algorithm 3.1, we also need to assign new weights to the edges connecting the regions in the new layer of the segmentation tree. For each pair of adjacent regions, we use the minimum weight of all edges across the region boundaries in the layer below. We also experimented with using the proximity classifier again on the larger regions or training an individual classifier for each layer in the segmentation tree. However, we found that all variants yield comparable performance and opted for the most efficient solution. As an example, three layers of our segmentation tree are visualized in Figure 3.4. In all our experiments, we construct the segmentation tree with a merging rate of  $p = 0.2$  until we obtain  $n_l = 10$  levels.

---

**Algorithm 3.1** Segmentation tree construction

---

**Input:** Superpixels, edges  $\mathcal{E}$

**Parameters:** Merging rate  $p$ , number of levels  $n_l$

Level  $l \leftarrow 1$

Regions in first level  $\mathcal{R}_l \leftarrow$  Superpixels

**while**  $l \leq n_l$  **do**

$l \leftarrow l + 1$

    Threshold  $w_{th} \leftarrow p$ th percentile of  $\text{weights}(\mathcal{E})$

    Merged edges  $\mathcal{E}_m \leftarrow \{e \mid e \in \mathcal{E}, \text{weight}(e) \leq w_{th}\}$

$\mathcal{R}_l \leftarrow \text{merge}(\mathcal{R}_{l-1}, \mathcal{E}_m)$

$\mathcal{E} \leftarrow \text{update}(\mathcal{E}, \mathcal{R}_l)$

**end while**

**Output:** Tree  $\{\mathcal{R}_l \mid l \in \{0 \dots n_l\}\}$

---

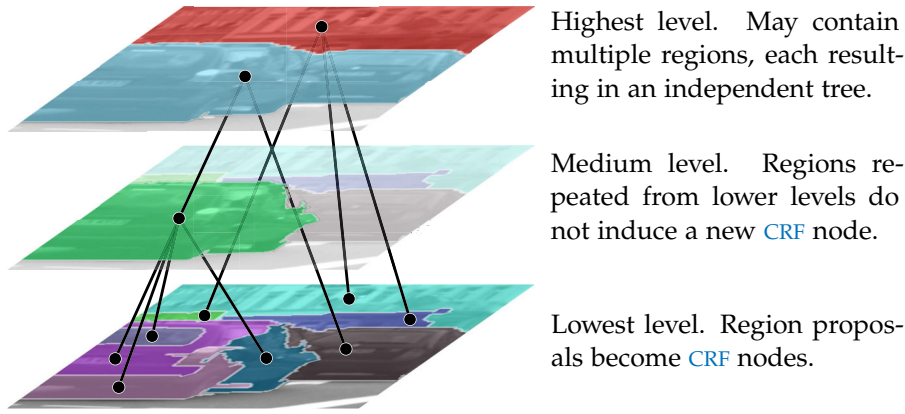


Figure 3.4: Segmentation tree and resulting CRF model. Regions are encoded by false colors, CRF nodes by black dots, and parent-child-relations by lines.

### 3.5 MULTI-CUE REGION CLASSIFICATION

Once the segmentation tree is computed, each segment is considered as a region proposal that is classified by three different classifiers. For each class of interest, the classifiers assign probability scores indicating the likelihood of the region being the respective class. Since our regions overlap and the classifiers might produce contradicting scores, the final pixel-level labels are inferred using a CRF as described in Section 3.6, where the scores serve as unary potentials. In order to maximize the efficiency of our system, all applied classifiers are based on features that can be efficiently computed for each region by exploiting the tree structure and accumulate the features from leaf nodes towards the root node.

The first two classifiers are responsible to classify regions based on their appearance and are based on the encode-and-classifier trees as described in Section 3.2. First, we average the RDF pixel classification results (c.f. Section 3.2.1) within the region, yielding  $s_{PC}$ . Second, we classify the aggregated texon histograms (c.f. Section 3.2.2) to obtain  $s_{TH}$ . Third, we propose a multi-cue region classifier providing  $s_{MC}$ . This classifier exploits features from the remaining input channels, c.f. Figure 3.1. Both, features and the classifier training are described in the following.

#### 3.5.1 Multi-cue features

Our features for multi-cue region classification describe the major properties of a proposal region that are discriminative between the different classes but are agnostic of the pixel-level texture. Such features include region geometry, as well as detector and point track information, an overview is provided in Table 3.2.

**REGION GEOMETRY** The first set of features aims to describe the shape and geometry of a region in both, 2D and 3D. The first feature  $G_1$  is the 2D aspect ratio of the bounding box around the region, which is for example helpful to discriminate upright pedestrians from elongated or square vehicles. Further, we allow the classifier to assess the type of boundary pixels. Buildings are often truncated by the image boundary, whereas objects such as pedestrians or vehicles are located on top of the ground. Therefore, we divide the set of region boundary pixels into four directions (top, bottom, left, right) with respect to the region center. For each direction, we count the number of boundary pixels that separate the region from another one ( $B_s$ ) or from the image border, *ground*, *sky* or *void* ( $B_v$ ), c.f. Figure 3.3 that contains superpixels with all such neighboring constellations. Then the ratio  $B_s/B_v+B_s$  yields the features  $G_2$ - $G_5$  for the four directions.

In street scenes, the height above ground is the most discriminative 3D feature, since different classes typically occur at different heights, e.g. vehicles and pedestrians vs. buildings and sky. Therefore, we define the height above ground of the top and bottom point of a 3D bounding box around the region as  $D_1$  and  $D_2$ , respectively. Note that we compute the bounding box extents by aggregating the robust superpixel bounding boxes that were obtained as described in Section 3.4.1. Additionally, we add the height, width, and depth of this 3D bounding box, as well as the center height above ground as features  $D_3$ - $D_6$ . The last set of geometric features deals with occlusions. While shape and region geometry are highly discriminative for fully visible objects, they might cause confusions when occlusions are present, as common in street scenes. Such occlusions cause the shape to alter and, for example, an occluded car might appear as a pedestrian. Thus, we compiled a collection of features that describes the occluded parts of the regions in order to allow for the classifier to leverage the full potential behind the remaining features. Using the same definitions for boundary pixels as above, we determine the depth jump in meters for all boundary pixels between object regions ( $B_s$ ). These values are then averaged for the four directions, yielding  $D_7$ - $D_{10}$ . In doing so, we avoid thresholding to determine the occlusions and allow for a learned interpretation that takes measurement noise into account. A large positive jump means that the boundary part is probably not occluded, while a negative value stands for an occlusion. A value around zero typically corresponds to an edge between close objects or a boundary within the same object. Note that we use the robust superpixel distances as described above to maximize the robustness of the occlusion features.

**DETECTORS AND POINT TRACKS** Detectors for ADAS typically operate at a high level of precision, i.e. regions that are covered by a detector bounding box most likely belong to the detector's class. How-

---

*2D region geometry*

G1	bounding box aspect ratio
G2-G5 <sup>a</sup>	there are two kind of boundary pixels: separating object regions ( $B_s$ ) or regions from image border, <i>ground</i> , <i>sky</i> or <i>void</i> ( $B_v$ ), c.f. Figure 3.3 that contains superpixels with all such neighboring constellations. This feature is defined as $B_s/B_v+B_s$

*Depth*

D1, D2	top (bottom) point height above ground
D3-D5	height (width, depth) [m]
D6	center height above ground [m]
D7-D10 <sup>a</sup>	occlusion strength: average depth jump (signed) at boundary pixels $B_s$

*Detectors*

DT1	overlap with vehicle detector bounding box
DT2	overlap with pedestrian detector bounding box

*Point tracks*

PT1	average velocity [ $\text{m s}^{-1}$ ]
-----	--

---

<sup>a</sup> Evaluated separately for boundary pixels located at top (right, bottom, left) relative to region center.

Table 3.2: Features for multi-cue region classification. Each feature is extracted on a region proposal of the segmentation tree.

ever, even these systems might produce false positives and typically have a rather low recall when objects are heavily occluded. Therefore, we add the detector information as features for our multi-cue classifier, such that it can combine this knowledge with the geometric and occlusion features. For each available detector  $i$ , we define  $DT_i$  as the overlap between the region and the detector's predicted bounding boxes.

The last multi-cue feature leverages the point track information. The speed of an object is a discriminative feature, since fast moving regions in street scenes typically contain vehicles. Thus, the feature  $PT_1$  is defined as the average velocity of the tracked points within the region. Note that we assign the median velocity of a superpixel to all tracks within that superpixel to obtain a more robust estimation, c.f. Section 3.4.1.

### 3.5.2 Training

Based on the multi-cue features describes in Section 3.5.1, we train a classifier to obtain a probability score for each class and region. As



with the superpixel proximity classifier (c.f. Section 3.4.2), we opt for a [RDF](#) due to the diversity of the features stemming from different domains with varying dynamic ranges. For training, we follow the same strategy and train until full depth with subsequent random pruning. Since regions from our segmentation tree might cover pixels with different ground truth labels, we need to cope with this effect during training. A simple solution would be to require the majority label within a region to cover at least 50 % of the pixels. However, in doing so, many training samples are discarded and information about the minority classes is lost. Instead, we assign the region’s feature vector to each pixel within the region and use these repeated vectors with the individual ground truth classes for training. As a side-effect, regions are weighted by their size, which causes the classifier to respect regions from higher levels of the segmentation tree, which are large but rare. Overall, we obtain a multi-cue classifier that tends to yield soft probability scores rather than hard decisions. These scores can then be fused well with the appearance classifiers via [CRF](#) inference, c.f. Section 3.6.

### 3.6 CRF INFERENCE

The output of the region classification step as described in Section 3.5, is our segmentation tree, where each region is associated with three sets of class scores. These scores represent the likelihoods of the regions belonging to the respective classes as estimated by the three classifiers described above. However, these scores might disagree and also the scores of overlapping regions in different levels of the segmentation tree might contradict each other.

In order to obtain a consistent labeling, we leverage a Conditional Random Field ([CRF](#)) to jointly infer the labels  $\mathbf{y} \in \mathcal{L}^n$  of all  $n$  regions in the segmentation tree. Each region corresponds to a node  $Y_i$  in the [CRF](#) with the label space  $\mathcal{L}$  consisting of the sought after classes plus an additional *void* label. The latter is introduced, since the regions might not belong to any of the considered classes or might contain a mixture of those. Therefore, we augment the classifier scores with a *void* label having an artificial score of  $|\mathcal{L}|^{-1}$ . Further, we normalize the scores such that they sum up to 1. If a parent region of the segmentation tree has a single child only, both, child and parent regions contain the same information. Thus, the parent node is excluded from the [CRF](#), see Figure 3.4. Note that in principle, one could use a Bayesian network as the hierarchy is a Directed Acyclic Graph ([DAG](#)) (Nowozin and Lampert, 2011), however, we opt for a [CRF](#) to allow for a direct integration of classifier scores as unary potentials.



Given all input data  $\mathbf{x}$ , the posterior probability is defined as

$$P(Y = \mathbf{y} \mid X = \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{i=1}^n \Phi_i(y_i) \prod_{(c,p) \in \mathcal{A}} \Psi_c(y_c, y_p) , \quad (3.7)$$

where  $\mathcal{A}$  denotes the set of parent-child relations as defined by the segmentation tree and  $Z(\mathbf{x})$  the partition function, c.f. Blake and Kohli, 2011; Nowozin and Lampert, 2011. In the following, the unary potentials  $\Phi_i$ , the parent-child potentials  $\Psi_c$ , and the inference scheme are described. Note that there is no potential connecting nodes within the same level of the segmentation tree, which renders Equation (3.7) tree-structured and allows for efficient inference as described in Section 3.6.3. For modeling smoothness priors between nodes in the segmentation tree, we solely rely on the parent-child relations as described in Section 3.6.2.

### 3.6.1 Unaries

As described in Section 3.5, for each region  $i$  in the segmentation tree, we obtain three different classifier scores, termed  $s_{PC}$ ,  $s_{TH}$ , and  $s_{MC}$ . A straightforward way to fuse these scores would be to use their product directly as the region's unary. Instead, we interpret the scores on the pixel level and compute the region likelihood as the product of all pixel scores. In doing so, larger parent regions are weighted up compared to their children nodes which is beneficial, since larger regions are typically more reliably classified due to a sufficient spatial support. Further, the parent-child relations, c.f. Section 3.6.2, are better balanced, since a parent region has the same influence as the sum of all children and a large child becomes more important than a smaller one.

Let  $n_i$  denote the number of pixels in region  $i$ . Then, the unary is defined as

$$\Phi_i(y_i) = \left( s_{PC}(y_i)^{\lambda_{PC}} s_{TH}(y_i)^{\lambda_{TH}} s_{MC}(y_i)^{\lambda_{MC}} \right)^{n_i} , \quad (3.8)$$

where the weights  $\lambda_{PC}$ ,  $\lambda_{TH}$ , and  $\lambda_{MC}$  capture the different reliabilities of the individual classifiers. These weights are optimized in initial experiments using grid search on the training set with the PASCAL VOC IoU score (Everingham et al., 2014) as objective function, c.f. Section 3.8.3.

### 3.6.2 Parent-child

Smoothness priors are expressed using factors between parent  $p$  and child nodes  $c$ , defined as

$$\Psi_c(y_c, y_p) = \begin{cases} 1 & \text{if } y_c = y_p \text{ or } y_p = \text{void} \\ e^{-n_c} & \text{otherwise} \end{cases} . \quad (3.9)$$

Again,  $n_c$  denotes the number of pixels in the child’s region. The influence of this factor is similar to Robust  $P^N$  potentials (Kohli et al., 2009) and expresses our expectation that parent and child nodes often belong to the same class. However, if a node is likely to contain multiple classes, it can be assigned to *void* and does not influence its child nodes anymore. Further, a small node may have a different label than the parent, even if the parent’s label is meaningful, e.g. a pedestrian in front of a large building.

### 3.6.3 Inference

The task of the inference step is to maximize the posterior probability mass function as described in Equation (3.7). However, instead of computing the maximizing labeling, we are interested in obtaining the marginal probabilities  $P(Y_i = y_i \mid \mathbf{X} = \mathbf{x})$  for all nodes  $i$  and apply a temporal filtering on top as described in Section 3.7.

Since Equation (3.7) is tree-structured, running sum-product belief propagation (Blake and Kohli, 2011; Nowozin and Lampert, 2011) is very efficient and provides the desired marginals without approximations. For all  $n_i$  pixels in a superpixel-level node  $i$ , we subsequently compute  $P(Y_i = y_i \mid \mathbf{X} = \mathbf{x})^{\frac{1}{n_i}}$ , normalize, and assign the result as marginal posteriors. The exponent  $\frac{1}{n_i}$  is applied to compensate for the weighting in Equation (3.8) and in turn resulting in a smoother probability mass function.

## 3.7 TEMPORAL REGULARIZATION

The inference mechanisms described so far exploited temporal information in form of point tracks via the multi-cue segmentation tree and classifier. However, the actual labeling was conducted individually for each frame and might be inconsistent over time. To overcome these limitations, we apply the time-recursive regularization scheme proposed in Scharwächter et al., 2014a.

The core idea is that pixels from different frames belonging to the same point track should ideally have the same semantic label. However, due to imperfect class predictions and erroneous tracks, label switches must be allowed, but should be punished. Therefore, Scharwächter et al., 2014a model these transitions using a Hidden Markov Model (HMM) filter associated with each point track.

In our system, we leverage the pixel-level CRF marginals and feed them to the filters in their update phase yielding temporally consistent posteriors. Since the point tracks are sparse, but we desire to filter all pixels in the image, we assign the filtered posteriors back to superpixels by averaging over all covered point tracks. The final labeling is then obtained as the label maximizing the marginal posterior, and assigned to all pixels in each superpixel.

### 3.8 EXPERIMENTS

In this section, we analyze our method in various experiments. We start by describing the actual algorithms that we used to obtain the input cues (Section 3.8.1). Subsequently, we introduce the datasets on which we conducted the experiments (Section 3.8.2) followed by a discussion of the evaluation metrics (Section 3.8.3). Next, we provide and discuss quantitative results (Section 3.8.4), conduct ablation studies to gain further insights into our method (Section 3.8.5), and provide exemplary result images (Section 3.8.6). We conclude by a detailed analysis of the run time of our system in Section 3.8.7.

#### 3.8.1 Cues

The cues depicted in Figure 3.1 (left column) are briefly outlined in the following. Note that pixel classification and textron histograms are already described in Sections 3.2.1 and 3.2.2, respectively. In principle, our overall approach is independent of the actual depth, motion or detector approaches used. Our choice for these experiments is mainly motivated by balancing the trade-off between quality and run time.

**DEPTH** To integrate depth information from stereo vision, we use dense disparity maps computed using Semi-Global Matching (SGM; Hirschmüller, 2008). Our implementation is based on the real-time Field-Programmable Gate Array (FPGA) adoption described in Gehrig et al., 2015.

**POINT TRACKS** Motion cues are integrated in terms of Kanade-Lucas-Tomasi (KLT) feature tracks (Tomasi and Kanade, 1991), which give a set of sparse but reliable long-term point trajectories. With the odometry information provided in the datasets, motion induced by camera movement is compensated using a Kalman filter, which provides an estimate of the 3D velocity of observed obstacles (Franke et al., 2005). Motion cues are used for grouping and classification of region proposals. Additionally, the tracks are also used to stabilize the final labeling over time, as described in Section 3.7.

**DETECTORS** Sliding-window object detectors have shown remarkable detection performance. At the same time, constraints on object and scene geometry help with computational efficiency. We employ object detectors for pedestrians and vehicles given that they are the most prominent dynamic objects in street scenes. We use a two-stage detection system for both classes, consisting of a fast Viola-Jones cascade (Viola and Jones, 2004) coupled with a three-layer convolutional neural network (Enzweiler and Gavrila, 2009) as an additional ver-

ification module. Multiple detections across location and scale are addressed using mean shift-based non-maximum suppression.

### 3.8.2 Datasets

We use two public datasets for our experimental evaluation, i.e. *Daimler Urban Segmentation* (DUS, Scharwächter et al., 2013) and *KITTI Vision Benchmark Suite* (Geiger et al., 2013). Both stem from urban street scenes that were recorded from a moving vehicle. As typical for such recordings and also as required by our method, both provide stereo image pairs and intermediate frames for motion cues. The Daimler Urban Segmentation (DUS) dataset contains 500 images with pixel-level semantic annotations of 5 different classes, i.e. *ground*, *vehicle*, *pedestrian*, *building*, and *sky*.

For KITTI, we collected annotations provided alongside previous publications (H. He and Upcroft, 2013; Ladický et al., 2014; Ros et al., 2015; Sengupta et al., 2013; C. Xu et al., 2013). Since the authors followed different label protocols, we mapped the individual labels to a common set of 6 classes, i.e. *ground*, *vehicle*, *pedestrian*, *building*, *vegetation*, and *sky*. We report numbers on all 216 annotated images provided for the visual odometry challenge and use the remaining 211 images for training.

Overall, the two datasets are comparably small, but nevertheless sufficiently large to train the RDEs utilized in this chapter. We turn towards large-scale datasets in Chapter 4.

### 3.8.3 Metrics

The PASCAL VOC IoU score as proposed in Everingham et al., 2014 is the standard metric for evaluating pixel-level semantic labeling and is also the metric that is used by the evaluation protocol for the DUS dataset. Thus, we consider this score as the major evaluation metric in our experiments. However, we argue that this metric alone is not sufficient to assess the quality of pixel-level labeling in street scenes and complement the evaluation with an object-centric evaluation. Both scores are motivated, defined, and analyzed in the following.

**PIXEL-LEVEL EVALUATION** The essential task of pixel-level semantic labeling is to assign a semantic class label to all pixels in the image. This suggests to consider each pixel in all the test images as an individual sample and to determine the correctness of classifying these samples. Since often an individual score for each semantic class is desirable, such metrics are typically computed for each class individually and then the overall performance is assessed as the average of the individual class scores. Especially, in street scenes where many image pixels contain ground and buildings and only a few pixels

		Prediction	
		true	false
Ground-truth	true	True Positive (TP)	False Negative (FN)
	false	False Positive (FP)	True Negative (TN)

Table 3.3: Binary confusion matrix.

belong to objects such as cars or pedestrians, this procedure has a beneficial side-effect, since it treats all classes equally, regardless of their size in the images. In the following, we will always follow this idea, which means that an overall metric refers to the average over the individual class performances without explicit mentioning.

For the moment, let us consider the classification of a single sample. In our case, this sample is an image pixel, but for other tasks this sample might be the whole image (image classification) or might be a bounding box proposal (object detection). Further, as discussed above, we assess the performance of a single class only, which reduces the analysis to a binary problem, where we assign the value *true* to the class of interest and *false* to all other classes. Now the ground truth and the predicted class can be either *true* or *false*, rendering four different cases that are summarized in Table 3.3. A *true* sample that is correctly classified is denoted as True Positive (TP) and as False Negative (FN) otherwise. A *false* sample that is erroneously classified as *true* is denoted as a False Positive (FP) and as True Negative (TN) otherwise. Note that in a multi-class setup, a confusion of two classes leads to a FP sample for the one class, and to a FN for the other. By accumulating over all test samples, we obtain the number of occurrences of these cases, which we denote by the respective abbreviations and the prefix  $\sum$ . Since in a multi-class setup these numbers need to be computed once for each considered class, a naive implementation would need to iterate over all test samples once for each class. Instead, it is more efficient to compute a multi-class confusion matrix first, and construct the desired numbers subsequently by summing the appropriate entries. Such a confusion matrix simply counts the number of occurrences of all pairs of ground truth and prediction labels.

Many widely used metrics for classification can be computed based on the definitions above (Fawcett, 2006). An example is the True Positive Rate (TPR), which is the proportion of *true* samples that are correctly classified, i.e.

$$\text{TPR} = \frac{\sum \text{TP}}{\sum \text{TP} + \sum \text{FN}}. \quad (3.10)$$

For our task of pixel-level semantic labeling a single metric that reflects TPs as well as both error cases FP and FN is desirable. Thus,

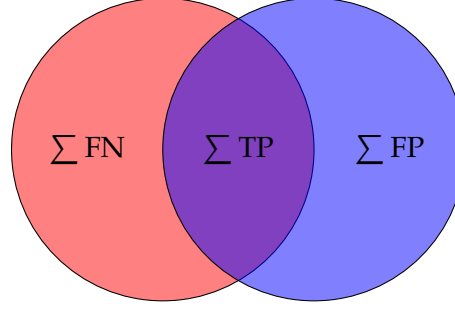


Figure 3.5: Overlapping schematic segments of ground truth (red) and prediction (blue) of same class and the resulting subsets  $\sum \text{TP}$ ,  $\sum \text{FP}$ , and  $\sum \text{FN}$ . The Intersection-over-Union (IoU) score is defined as the overlap of ground truth and prediction, i.e. the intersection ( $\sum \text{TP}$ ) over the union ( $\sum \text{TP} + \sum \text{FP} + \sum \text{FN}$ ).

Everingham et al., 2014 introduce the Intersection-over-Union (IoU) metric, which is defined as

$$\text{IoU} = \frac{\sum \text{TP}}{\sum \text{TP} + \sum \text{FP} + \sum \text{FN}}. \quad (3.11)$$

The term *Intersection-over-Union* becomes clear, when we interpret the pixel-level semantic labeling from a segmentation point of view, as emphasized by *semantic segmentation*, an equivalent term for pixel-level semantic labeling. If we consider connected regions in the prediction and ground truth images with identical labels as segments,  $\sum \text{TP}$ ,  $\sum \text{FP}$ , and  $\sum \text{FN}$  can be defined as the areas of different parts of these segments, c.f. Figure 3.5. Then, the IoU metric is the size of the intersection between prediction and ground truth over the union of both segments and hence its name. Note that the size of such a segment is computed as the number of covered pixels. An alternative name that is sometimes used in literature is Jaccard Index (JI), which stems from the analysis of sets. Note that in many datasets, some pixels are not annotated in the ground truth or carry the label *void*. Throughout this work, we exclude such pixels from evaluation.

**OBJECT-CENTRIC EVALUATION** The IoU score introduced above is based on counting pixels. Therefore, we do not consider this score alone to be sufficient for the high range of scales of objects in urban scenes. The metric is dominated by objects at a large scale, whereas smaller ones have only a minor impact. However, for most practical applications, a metric that answers the question of how well individual objects are represented in the scene is desirable. Therefore, we complement the IoU score with an object-centric evaluation.

To obtain such a metric, we divided the *vehicle* and *pedestrian* annotations for the DUS dataset into individual instance labels. Since our work aims to perform pixel-level labeling, it does not produce object instance predictions. Yet, we argue that it is beneficial to assess the semantic labeling performance by taking instances into account.

To that end, we define an evaluation protocol, where we create artificial instances given the pixel-level labeling output, see Figure 3.6 for an example and Algorithm 3.2 for details. Each predicted pixel is assigned to the closest ground truth instance with matching label. For this step, we only consider ground truth instances that overlap with the predicted *semantic segment*. This segment is defined as the connected component with constant label that the predicted pixel belongs to. If no ground truth candidate is available, the whole semantic segment is considered a FP instance. These artificially created instances are then considered as TP<sub>s</sub> if they overlap with their matching ground truth instances by at least 50 % and as FP<sub>s</sub> otherwise. Note that the definition of the overlap equals the IoU definition according to Figure 3.5. Unmatched ground truth instances count as FN<sub>s</sub>. Overall, this procedure allows for an interpretation of the pixel-level output in terms of TP, FP or FN instances. In other words, instead of considering pixels as our test samples as with the IoU score, we consider whole object instances. In doing so, the contribution of per-pixel errors is normalized with the scale of the instance and the evaluation protocol allows instance-aware evaluation without instance-level predictions.

Based on these instance-level counts, we could again compute the IoU value. However, we use a slightly altered metric named F-score, which is more common in the context of object detection and defined as

$$F = \frac{2 \sum TP}{2 \sum TP + \sum FP + \sum FN} . \quad (3.12)$$

This metric is related to the AP<sup>r</sup> score from Hariharan et al., 2014. The major differences are that in Hariharan et al., 2014 real instance predictions are evaluated, while we use artificially created ones during our evaluation protocol. Further, Hariharan et al., 2014 compute a precision-recall curve by varying a threshold on the prediction likelihood. Since we do not have such likelihoods here, we are restricted to a single operating point on this curve, for which we report the F-score.

#### 3.8.4 Comparison to baselines

On the DUS dataset, we compare our method to six baselines. The authors either published their performance (M.-Y. Liu et al., 2015; Scharwächter et al., 2014a; Sharma et al., 2015) or have code available (Gould, 2012; Ladický et al., 2010; Shotton et al., 2008). For Gould, 2012, we use the pairwise results from the *multiSeg* example. The numbers for Shotton et al., 2008 are generated with C# code provided by Matthew Johnson and represent the final result of their two step approach. Compared to this baseline, which is closely related to the encode-and-classify concept, we significantly improve labeling accuracy. For all experiments that we conduct with public software,



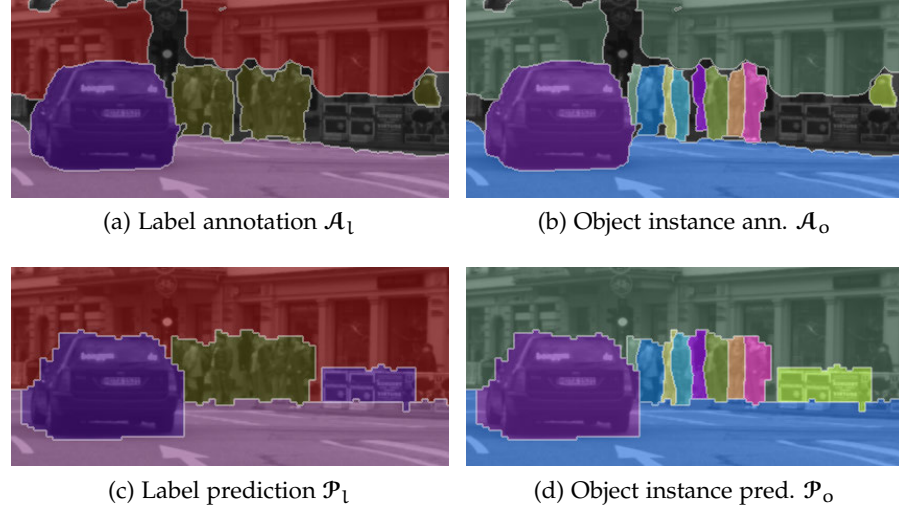


Figure 3.6: Overview of different annotation and prediction formats. Object instance prediction (d) is generated from Algorithm 3.2. Note the false positive *vehicle* prediction to the right of the pedestrians.

we use the default parameters set in the code. Scharwächter et al., 2014a; Sharma et al., 2015 kindly provided their inferred results to allow for an evaluation with our novel object-centric metric. For the KITTI dataset, there are no reported results available that make use of all annotated images. Therefore, we only compare to Gould, 2012; Ladický et al., 2010; Shotton et al., 2008.

Tables 3.4 to 3.7 indicate that our approach is competitive compared to the baselines on both datasets regarding semantic labeling quality. In particular, traffic participants (*vehicle*, *pedestrian*) are recognized with superior performance in terms of pixel and object accuracy, mostly as a result of incorporating the additional object detectors at multiple levels in our system; on KITTI our approach outperforms the baselines by a margin of 30 %. We further observe that the results based on Stixels outperform GBIS, which in turn outperform those based on SEOF. We attribute this to the main properties of the individual superpixel variants. Stixels are specifically designed for street scenes, whereas GBIS and SEOF are more generic. Further, GBIS has no restrictions on the superpixel shape, while SEOF has a strong compactness prior. The latter is especially disadvantageous for objects at larger distances as indicated by the F-score.

In addition to those three superpixel variants, we also show results when using the gPb-OWT-UCM segmentation tree from Arbeláez et al., 2011. This method does not take depth into account, but is still highly competitive and has been used as the basis for the segmentation trees by Silberman et al., 2012 and S. Gupta et al., 2014, which we consider as related to our approach. For the UCM SPS column in Table 3.4, we chose a single threshold in the UCM hierarchy to extract superpixels with a size similar to the other superpixel variants and compute our



	IoU <sup>a</sup>	IoU <sup>b</sup>	F <sup>b</sup>	rt <sup>c</sup>
<i>Baselines</i>				
Shotton et al., 2008	70.1	58.3	53.3	125
Gould, 2012	73.5	44.9	65.1	5200
Scharwächter et al., 2014a	80.6	72.4	81.4	150 <sup>d</sup>
Sharma et al., 2015	84.5	73.8	85.1	2800
Ladický et al., 2010	86.0	74.5	83.8	10 <sup>5</sup>
M.-Y. Liu et al., 2015	<b>86.3</b>	77.2	-	<b>110</b>
<i>Our multi-cue system</i>				
UCM tree	81.8	79.6	82.8	10 <sup>4</sup>
UCM SPS	80.7	<b>81.0</b>	79.7	10 <sup>4</sup>
SEOF	83.5	78.4	79.6	10 <sup>4</sup>
GBIS	84.3	79.1	81.2	410
Stixels	85.7	79.9	<b>86.4</b>	163
<i>Cues removed (Stixels)</i>				
-DT	84.2	75.9	82.7	118
-PT	84.9	78.8	84.5	138
-DT, -PT	82.6	73.2	80.4	93

<sup>a</sup> Average over all classes

<sup>b</sup> Average over dynamic objects: *vehicle, pedestrian*

<sup>c</sup> Run time in ms

<sup>d</sup> This also includes computation time for stereo and Stixels, which are neglected in the run times reported in Scharwächter et al., 2014a.

Table 3.4: Averaged quantitative results on the DUS dataset with official train/test split compared to six baselines. We show pixel accuracy (IoU), the F-score to assess object accuracy (F), and run time (rt). IoU and F-score are given in percent, the run time in ms. Additionally, we report results using Stixels, where detections (-DT), point tracks (-PT), and their combination (-DT, -PT) are removed from the full system to demonstrate their influence on the overall performance.

---

**Algorithm 3.2** Generate object instances from label prediction. These are used for an evaluation at the instance level.

---

**Input:** Label prediction  $\mathcal{P}_l$ , Object instance annotation  $\mathcal{A}_o$

Allocate searched object instance prediction  $\mathcal{P}_o$

$\mathcal{R}_p \leftarrow \text{connectedRegions}(\mathcal{P}_l)$

**for all** regions  $R \in \mathcal{R}_p$  **do**

$l \leftarrow \text{label}(R)$

    // Find all candidate object instances for the region:

$\mathcal{C}_v \leftarrow \{v \mid v \in \mathcal{A}_o(R) \wedge \text{label}(v) = l\}$

**if**  $\mathcal{C}_v = \emptyset$  **then**

$\mathcal{P}_o(R) \leftarrow \text{new object instance (false positive)}$

**else**

**for all** pixels  $p \in R$  **do**

            // Find closest candidate object instance for p:

$\mathcal{P}_o(p) \leftarrow \underset{v \in \mathcal{C}_v}{\text{argmin}} \min_{\substack{\text{pixel } q \in \mathcal{A}_o, \text{ with} \\ \text{label}(q) = \text{label}(v)}} \text{dist}(p, q)$

**end for**

**end if**

**end for**

**Output:**  $\mathcal{P}_o$

---

	IoU <sup>a</sup> [%]	IoU <sup>b</sup> [%]
<i>Baselines</i>		
Shotton et al., 2008	53.7	27.1
Gould, 2012	70.7	41.9
Ladický et al., 2010	73.9	50.4
<i>Our multi-cue system</i>		
Stixels	<b>75.8</b>	<b>65.2</b>

<sup>a</sup> Average over all classes

<sup>b</sup> Average over dynamic objects: *vehicle, pedestrian*

Table 3.5: Averaged quantitative results on the [KITTI](#) dataset compared to three baselines.

	IoU					F-score	
	<i>veh</i>	<i>ped</i>	<i>gro</i>	<i>bui</i>	<i>sky</i>	<i>veh</i>	<i>ped</i>
<i>Baselines</i>							
Shotton et al., 2008	70.5	46.1	93.6	73.5	66.6	67.4	39.3
Gould, 2012	68.7	21.3	95.7	87.6	94.2	87.5	42.7
Scharwächter et al., 2014a	78.9	65.9	93.8	89.2	75.4	85.1	77.7
Sharma et al., 2015	79.3	68.4	96.7	86.3	91.4	90.8	79.4
Ladický et al., 2010	76.0	73.0	94.9	90.7	<b>95.5</b>	86.0	81.7
M.-Y. Liu et al., 2015	83.3	71.1	96.4	<b>91.2</b>	89.5	-	-
<i>This paper</i>							
UCM tree	84.7	74.4	<b>96.8</b>	84.1	69.0	85.3	80.3
UCM SPS	<b>86.5</b>	75.5	<b>96.8</b>	86.2	58.8	86.5	72.9
SEOF	81.6	75.2	88.2	88.9	83.9	81.6	77.5
GBIS	81.4	<b>76.9</b>	87.6	89.4	85.9	83.7	78.7
Stixels	85.4	74.3	96.2	89.6	82.8	<b>90.9</b>	<b>82.0</b>
<i>Cues removed (Stixels)</i>							
-DT	82.9	68.9	96.2	89.6	83.3	90.8	74.6
-PT	85.0	72.6	96.2	89.5	81.2	90.8	78.2
-DT, -PT	81.6	64.8	96.2	88.9	81.7	90.0	70.8

Table 3.6: Quantitative results on the [DUS](#) dataset with official train/test split compared to six baselines for the classes *vehicle*, *pedestrian*, *ground*, *building*, and *sky*. Additionally, we report results using Stixels, where detections (-DT), point tracks (-PT), and their combination (-DT, -PT) are removed from the full system to demonstrate their influence on the overall performance. All numbers are in percent.

hierarchy on top of it. For the [UCM tree](#) column, we chose 10 representative thresholds to remove our hierarchy generation completely from the system and use the [UCM tree](#) instead. We find that performance drops when using the [UCM tree](#), which shows the benefit of our proposed multi-cue segmentation tree.

### 3.8.5 Ablation studies

For a second experiment, we limit ourselves to using Stixels as super-pixels and evaluate the contributions of the external cues introduced in Section 3.8.1, i.e. detectors (-DT) and point tracks (-PT). If a single cue is omitted, the overall performance decreases, but our method is still competitive to all baselines, c.f. Table 3.4. As soon as both cues are left out, performance drops more significantly.

In order to evaluate the role of the specific object detector chosen, we replaced our detector (Enzweiler and Gavrila, 2009) that is specialized for street scenes with a state-of-the-art general purpose detector, namely Faster [R-CNN](#) as proposed in S. Ren et al., 2015. We

	<i>veh</i>	<i>ped</i>	<i>gro</i>	<i>bui</i>	<i>veg</i>	<i>sky</i>
<i>Baselines</i>						
Shotton et al., 2008	50.6	3.6	79.4	63.5	71.0	54.3
Gould, 2012	75.8	8.0	89.8	83.4	85.8	<b>81.6</b>
Ladický et al., 2010	76.0	24.8	<b>89.9</b>	<b>84.9</b>	<b>86.6</b>	81.2
<i>This paper</i>						
Stixels	<b>79.1</b>	<b>51.2</b>	89.7	79.6	77.7	77.5

Table 3.7: Quantitative results on the [KITTI](#) dataset compared to three baselines for the classes *vehicle*, *pedestrian*, *ground*, *building*, *vegetation*, and *sky*. All numbers denote the [IoU](#) score of the respective class in percent.

repeated the experiments of our full system using Stixels as superpixels and the [R-CNN](#) detector to provide the detector cues. The overall performance is 78.2 % avg. [IoU](#) of dynamic objects, which lies in between of the performance without a detector (75.9 %) and with our specialized detector (79.9 %), c.f. Table 3.4. Further, the run time of the Faster [R-CNN](#) detector is 110 ms slower than our standard detector. We conclude that our system efficiently exploits the information provided by the object detector, it naturally depends on its performance, but nevertheless achieves decent results even in the absence of such a detector.

### 3.8.6 Qualitative results

Figure 3.7 shows example labeling results of our approach on the [DUS](#) dataset versus the baseline (Ladický et al., 2010) and ground truth. Note that for all three superpixel variants our method produces competitive results compared to the baseline while being orders of magnitude faster. We further observe that our multi-cue approach helps to overcome problems with 3D object geometry and scene structure that exist in Ladický et al., 2010. This demonstrates the strength of consistently exploiting multiple cues, which is an integral part of our approach. Note further that the [GBIS](#) variant is the best performing method for the class *pedestrian*, while the Stixel variant is the most accurate for the class *ground*. These qualitative observations match the quantitative results in Table 3.6.

### 3.8.7 Run time

In this section, we address the run time of our multi-cue approach as real-time speeds are crucial for autonomous driving, c.f. Section 1.3. We evaluate run time using a system consisting of an Intel Core i7 [CPU](#) and an NVIDIA GTX 770 [GPU](#). First, we report the time needed

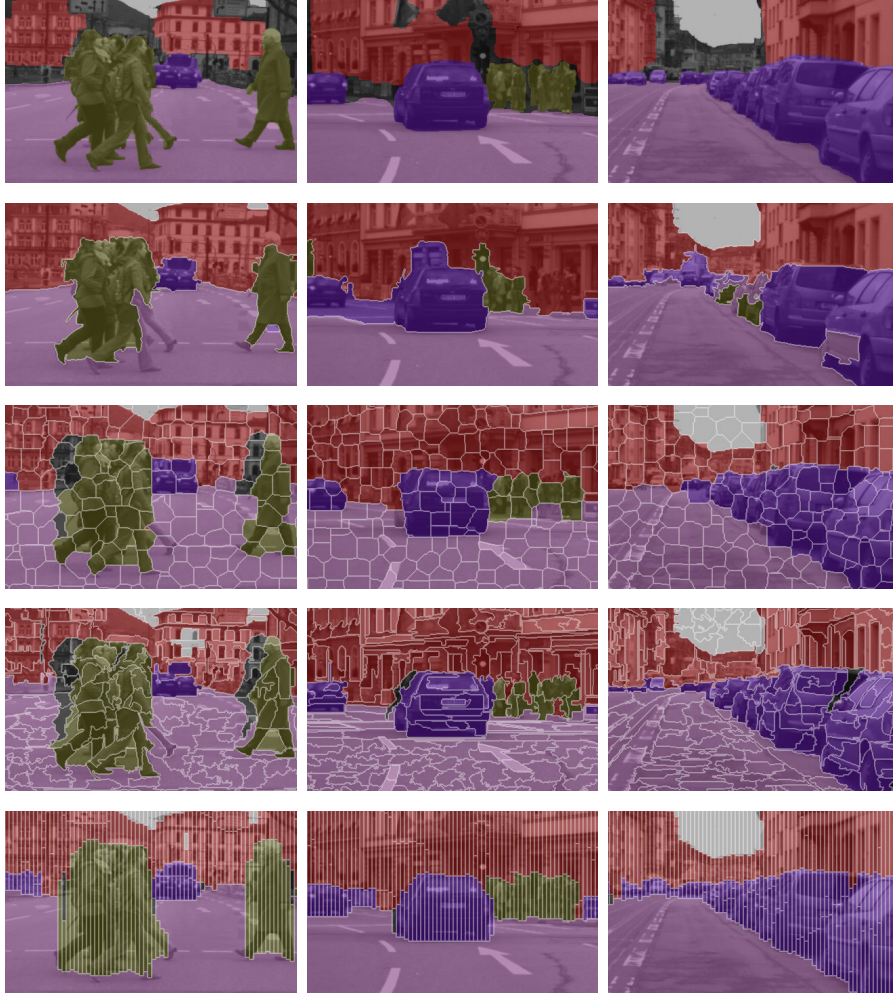


Figure 3.7: Qualitative results on the [DUS](#) dataset compared to the ground truth and a baseline. From top to bottom: ground truth, baseline (Ladický et al., 2010), and our results for [SEOF](#), [GBIS](#) and Stixels superpixels.

Run time of individual components in ms

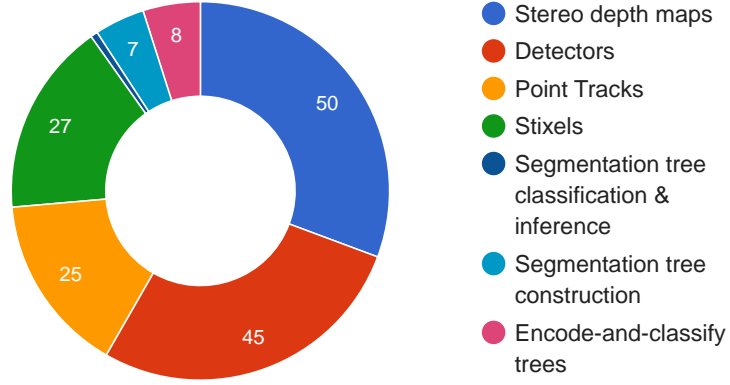


Figure 3.8: Breakdown of run time. We show the contribution of the individual components of our overall system to the 163 ms run time reported with Stixel superpixels in Table 3.4. About 95 % of the run time accounts to the inputs to our system, out of which Stixels, stereo, detectors, and point-tracks account for 90 % of the run time and are typically readily available in an intelligent vehicle (Franke et al., 2013). Our approach centered around the multi-cue segmentation tree needs only little run time on top to produce high-quality semantic labeling results, clearly indicating its efficiency. Courtesy of Timo Rehfeld.

to parse a single image of the [DUS](#) dataset in Table 3.4. Note that this overall parsing time includes the computation of all input cues. For depth, we assume 50 ms run time as reported for the dataset in Scharwächter et al., 2013. As the reference implementation of Shotton et al., 2008 is not tuned for fast execution, we quote the optimized timing results originally reported in Shotton et al., 2008. In summary, our system has a faster parsing time compared to most baselines. Scharwächter et al., 2014a and Shotton et al., 2008 are slightly faster, but at the cost of a large drop in classification quality; M.-Y. Liu et al., 2015 is competitive in terms of both, run time and accuracy.

Next, we analyze the run time of our method with Stixel superpixels in more detail to demonstrate the efficiency of the multi-cue models we focus on in this work. To this end, we break down the overall system run time into its individual components, c.f. Figure 3.8. The proposed multi-cue segmentation tree is very efficient and requires only less than 5 % of the overall run time, including construction, classification, and inference. Especially, when assuming the remaining components (Stixels, stereo, detectors, and point-tracks) as given, c.f. Franke et al., 2013, we add only little overhead to obtain a high-quality semantic labeling output.

Since in an autonomous vehicle, there are typically multiple computational resources available (Franke et al., 2013), the execution of the individual components can be pipelined, i.e. while the current frame is still processed, computations on the next frame are already

started using different computational resources. In this work, we run stereo depth maps and detectors in parallel on individual parts of an [FPGA](#), point tracks and encode-and-classify trees are computed on the [GPU](#), as well as Stixels and the multi-cue segmentation tree on the [CPU](#). In doing so, no resource is occupied for more than 50 ms per frame, yielding a throughput of 20 fps at the cost of one frame delay.

Conceptually, our approach combines the central ideas from the two run time efficient methods (Scharwächter et al., 2014a; Shotton et al., 2008) and extends them with tree-structured [CRF](#) inference, multiple cues and the encode-and-classify concept. In this way, we are able to significantly improve labeling accuracy compared to these methods, while maintaining fast inference time.

### 3.9 DISCUSSION

In this chapter, we proposed and analyzed a system for urban pixel-level semantic labeling. Our system is built on top of various components that are readily available in intelligent vehicles and are established and tested over many years. We opted for a standard processing pipeline that we altered such that all available cues are exploited to their full extent throughout all system components. Central to this concept is the multi-cue segmentation tree that allows for efficient construction, feature aggregation, classification, and inference. The resulting system achieves high labeling performance while being computationally efficient at the same time. Most prominently, we increase the labeling performance for the two most relevant and challenging classes (vehicles and pedestrians) by a significant margin over the state of the art on the considered datasets despite near real-time speeds. We take this as evidence for the suitability of our segmentation tree and the holistic integration of multiple complementary cues, particularly from the perspective of real-time applications. The proposed concept is independent of the actual input data and can deal well with different superpixels, detectors or even with completely missing input data. Best performance is obtained, when all available cues are used and when the system is built on Stixels, which are superpixels designed for urban scenes. Overall, our multi-cue approach can be easily added to existing components for autonomous driving and achieves high-quality semantic labelings.

However, our method strongly depends on the accuracy of its multi-cue input data. It is designed to maximally exploit the information available in its inputs by combining the strengths of various cues, but it does not have direct access to the input images at a pixel-level. We can barely recover from errors present in this data, e.g. under-segmentations of the superpixels, misclassifications of the encode-and-classify trees or detectors, or noise in stereo and point tracks. Therefore, we next consider methods from the field of deep learning

that directly operate on the input pixels. These methods are capable of jointly learning features for scene representation, incorporate context knowledge, and perform classification. While our system needs only a few images for training, deep learning methods typically require a tremendous amount of training data. Thus, the following chapter introduces the Cityscapes dataset and analyzes the potential behind such a large-scale dataset.



## LARGE-SCALE SEMANTIC SCENE UNDERSTANDING

---

### CONTENTS

4.1	Introduction . . . . .	57
4.2	Dataset . . . . .	58
4.2.1	Data specifications . . . . .	59
4.2.2	Classes and annotations . . . . .	61
4.2.3	Dataset splits . . . . .	64
4.2.4	Statistical analysis . . . . .	66
4.3	Pixel-level Semantic Labeling . . . . .	72
4.3.1	Metrics . . . . .	73
4.3.2	Control experiments . . . . .	74
4.3.3	Baselines . . . . .	75
4.3.4	Pixel-level benchmark . . . . .	81
4.3.5	Real-time fully convolutional networks . . . . .	94
4.3.6	Cross-dataset evaluation . . . . .	98
4.4	Instance-level Semantic Labeling . . . . .	99
4.4.1	Metrics . . . . .	99
4.5	Benchmark Suite . . . . .	101
4.6	Impact and Discussion . . . . .	102

---

In the previous chapter, we investigated the potential of leveraging readily available multi-cue input data for semantic urban scene understanding. We saw that the use of multiple cues is indeed beneficial, but that the performance of the analyzed system was limited by the accuracy of available input cues. We expect these cues, in turn, to be limited by the amount of available training data, especially the performance of appearance-based classification. Further, both datasets used were recorded in a single city and have rather correlated images stemming from a few video streams only. Thus, the classifiers are likely overfitted to the respective dataset they were trained on and the generalization to real-world applications remains unclear.

In this chapter, we follow a different path and investigate methods for large-scale urban semantic scene understanding. Object recognition tasks, such as image classification or object detection, have benefited enormously from large-scale datasets, especially in the context of deep learning. For semantic urban scene understanding, however, no current dataset adequately captures the complexity of real-world urban scenes.



(a) train/val – fine annotation – 3475 images



(b) train – coarse annotation – 20 000 images



(c) test – fine annotation – 1525 images

Figure 4.1: Example images from the Cityscapes dataset with annotations as false-color overlays. Overall, the dataset includes 5000 images with fine and 20 000 images with coarse annotations.

To address this, we introduce *Cityscapes*<sup>1</sup>, a large-scale dataset and benchmark suite to train and evaluate methods for pixel-level and instance-level semantic labeling. Cityscapes is comprised of video sequences recorded with a stereo camera in streets from 50 different cities to achieve a high level of diversity. We provide high quality pixel-level annotations for 5000 images of these recordings and coarse annotations for 20 000 additional images to facilitate methods that exploit large-scale weakly-labeled data. In doing so, we exceed previous efforts in terms of dataset size, annotation quality and richness, diversity, and scene complexity. Based on our benchmark, we can compare the performance of state-of-the-art approaches and analyze their suitability for applications in autonomous driving. Example images of the Cityscapes dataset can be found in Figure 4.1.

<sup>1</sup> [www.cityscapes-dataset.net](http://www.cityscapes-dataset.net)

The remainder of this chapter is structured as follows. We start with an introduction in Section 4.1, where we present related datasets and motivate the need for a new large-scale dataset. Subsequently, we provide a detailed description of the dataset as well as an in-depth analysis of the dataset characteristics (Section 4.2). The accompanying benchmark suite focuses on two major challenges, i.e. *pixel-level semantic labeling* and *instance-level semantic labeling*. The first is discussed in Section 4.3, where we introduce the task, gain further insights via control and oracle experiments, and analyze state-of-the-art methods. Subsequently, we briefly introduce the instance-level task in Section 4.4. Eventually, we conclude with a discussion of the benchmark (Section 4.5) and its impact on the research community (Section 4.6).

This chapter is largely based on Cordts et al., 2016 as well as Cordts et al., 2015 and contains verbatim quotes of the first work. Sections in this dissertation that share contributions with other authors are marked via footnotes.

#### 4.1 INTRODUCTION<sup>2</sup>

The resurrection of deep learning (LeCun et al., 2015) has had a major impact on the current state of the art in machine learning and computer vision. Many top-performing methods in a variety of applications are nowadays built around deep neural networks, c.f. Section 2.1.2. A selection of works with a high impact is Krizhevsky et al., 2012 for image classification, Shelhamer et al., 2017 for pixel-level semantic labeling, and Girshick, 2015 for object detection. A major contributing factor to their success is the availability of large-scale, publicly available datasets such as *ImageNet/ILSVRC* (Russakovsky et al., 2015), *PASCAL VOC* (Everingham et al., 2014), *PASCAL-Context* (Motlaghi et al., 2014), and *Microsoft COCO* (T.-Y. Lin et al., 2014) that allow deep neural networks to develop their full potential. Despite the existing gap to human performance, scene understanding approaches have started to become essential components of advanced real-world systems.

The application of self-driving cars is becoming increasingly popular, but is also particularly challenging and makes extreme demands on system performance and reliability. Consequently, significant research efforts have gone into new vision technologies for understanding complex traffic scenes and driving scenarios, e.g. Badrinarayanan et al., 2017; Franke et al., 2013; Furgale et al., 2013; Geiger et al., 2014; Ros et al., 2015; Scharwächter et al., 2014a or the multi-cue system that we described in Chapter 3. Also in this area, research progress can be heavily linked to the existence of datasets such as

<sup>2</sup> This section contains verbatim quotes of Section 1 of the publication Cordts et al., 2016. Note that all authors of that work contributed to this section.

the *KITTI Vision Benchmark Suite* (Geiger et al., 2013), *CamVid* (Brostow et al., 2009), *Leuven* (Leibe et al., 2007), and *Daimler Urban Segmentation* (DUS, Scharwächter et al., 2013). These urban scene datasets are often much smaller than datasets addressing more general settings. Moreover, we argue that they do not fully capture the variability and complexity of real-world inner-city traffic scenes. Both shortcomings currently inhibit further progress in visual understanding of street scenes. To this end, we propose the *Cityscapes* benchmark suite and corresponding dataset, specifically tailored for autonomous driving in an urban environment and involving a much wider range of highly complex inner-city street scenes that were recorded in 50 different cities. Cityscapes significantly exceeds previous efforts in terms of size, annotation richness, and, more importantly, regarding scene complexity and variability. Despite the main focus of this dissertation, we go beyond pixel-level semantic labeling by also considering instance-level semantic labeling in both our annotations and evaluation metrics. To facilitate research on 3D and multi-cue scene understanding, we also provide depth information through stereo vision.

The research goals of this chapter are two-fold. First, we focus on the Cityscapes dataset itself. We introduce its main properties and discuss their importance for a suitable dataset in the context of urban scene understanding. Further, we analyze the main characteristics, compare them with related datasets, and show that Cityscapes has the potential to establish a new reference for training and testing of algorithms in the field of autonomous driving. As a second research goal, we benchmark the performance of various methods for pixel-level semantic scene understanding. In doing so, we can assess these methods in terms of their suitability for self-driving cars. We can use this knowledge to understand the properties of different approaches and to derive important design choices.

## 4.2 DATASET<sup>3</sup>

Designing a large-scale dataset requires a multitude of decisions, e.g. on the modalities of data recording, data preparation, and the annotation protocol. Our choices were guided by the ultimate goal of enabling significant progress in the field of semantic urban scene understanding. We discuss and justify these guidelines in this section and analyze the characteristics of the Cityscapes dataset.

<sup>3</sup> The creation of the Cityscapes dataset was initially driven by Marius Cordts and was then continued as joint work with Mohamed Omran. The analyses in this section are based on joint work of Mohamed Omran and Marius Cordts in discussions with the other authors of the publication Cordts et al., 2016. This section contains verbatim quotes of Section 2 in that publication.

#### 4.2.1 Data specifications

Our data recording and annotation methodology was carefully designed to capture the high variability of outdoor street scenes. Several hundreds of thousands of frames were acquired from a moving vehicle during the span of several months, covering spring, summer, and fall in 50 cities, primarily in Germany but also in neighboring countries. We deliberately did not record in adverse weather conditions, such as heavy rain or snow, as we believe such conditions to require specialized techniques and datasets (Pfeiffer et al., 2013).

Our camera system and post-processing reflect the state of the art in the automotive domain. Images were recorded with an automotive-grade 22 cm baseline stereo camera using  $\frac{1}{3}$  in CMOS 2 MP sensors (OnSemi AR0331) with rolling shutters at a frame-rate of 17 Hz. The sensors were mounted behind the windshield and yield HDR images with 16 bits linear color depth. In contrast to DSLR or smartphone cameras as used for many large scale datasets (Everingham et al., 2014; T.-Y. Lin et al., 2014; Russakovsky et al., 2015), such automotive cameras have quite opposing requirements. The focus is to record realistic rather than visually pleasant images with over-saturated colors, since the purpose of such cameras is to enable environment perception for the machine and not for the human. Further, automotive cameras are facing extreme environment conditions, e.g. with temperatures that might range from  $-40^{\circ}\text{C}$  to  $125^{\circ}\text{C}$ . Simultaneously, the cameras must have a long lifetime while having only low costs. Overall, these requirements cause our images to be visually different from comparable datasets and support the need for such a dataset recorded with suitable sensors for self-driving cars.

Each 16 bit stereo image pair was debayered and rectified after recording. We relied on Krüger et al., 2004 for extrinsic and intrinsic calibration. To ensure calibration accuracy we re-calibrated on-site before each recording session. For comparability and compatibility with existing datasets we also provide Low Dynamic-Range (LDR) 8 bit RGB images that are obtained by applying a logarithmic compression curve. Such mappings are common in automotive vision, since they can be computed efficiently and independently for each pixel. To facilitate highest annotation quality, we applied a separate tone mapping to each image. The resulting images are less realistic, computationally expensive to compute, and in-consistent between left and right image of a stereo pair, which renders them unsuitable for an automotive application. However, they are visually more pleasing and proved easier to annotate. Examples of different variants of preprocessing a 16 bit image are shown in Figure 4.2. The examples unveil the large amount of information present in HDR images, especially when facing challenging illumination conditions.

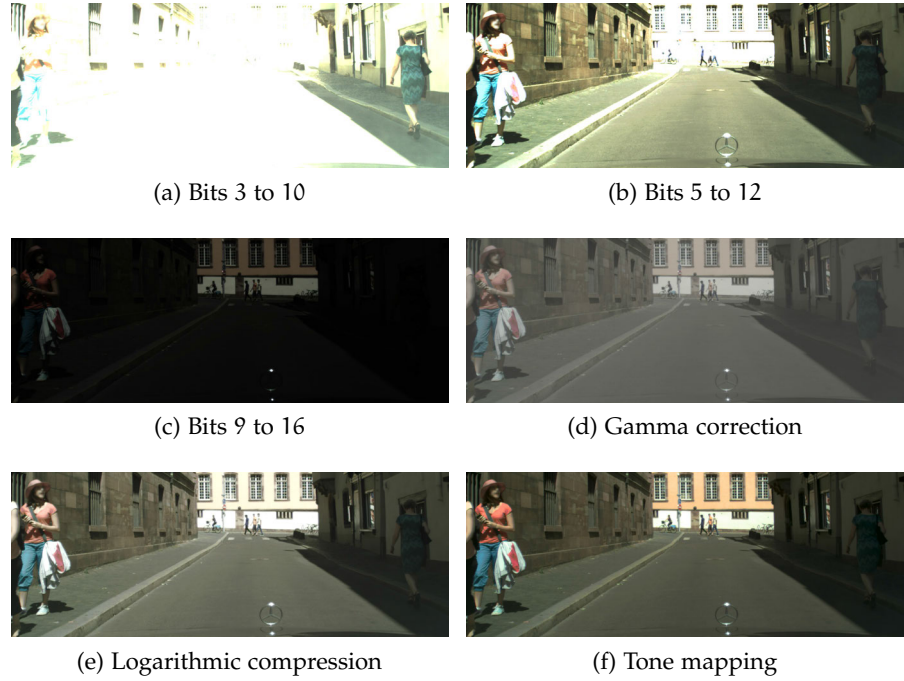


Figure 4.2: Different variants of preprocessing of a 16 bit image from our automotive camera. To visualize the information content of this image we clip to different 8 bit sub-ranges (a)–(c). For the published automotive-capable images within this dataset, we applied a logarithmic compression curve (e). For comparison, we show a gamma correction applied to the same input image (d). To facilitate accurate annotations, we leveraged a tone mapping (f).



Out of the video recordings, we selected images for annotation, see Section 4.2.2 for details on the labeling process and Figure 4.1 for examples. Since annotations are very costly and time-consuming to obtain, we sought after images with a high diversity of foreground objects, background, and overall scene layout. To achieve these requirements, we conducted the selection manually and chose 5000 images from 27 cities for dense pixel-level annotation. In practice, we pressed a steering-wheel button during the recordings to mark candidate images. For each such annotated image, we provide a 30-frame video snippet in full, where the annotated image is the 20<sup>th</sup> frame. In doing so, we can supply context information from the time domain. Causal methods can leverage 20 frames prior to the one of interest, such that trackers and filters can converge. A-causal methods, e.g. video segmentation, have access to the whole 30 frame snippet. In addition, we include the whole recording drive from one city of our validation set (c.f. Section 4.2.3), such that methods requiring a longer time span at test but not training time can be evaluated. For the remaining 23 cities, a single image every 20 s or 20 m driving distance (whatever comes first) was selected for coarse annotation, yielding 20 000 images in total. Note that the two triggers are on par for an average driving speed of  $3.6 \text{ km h}^{-1}$ , thus typically an image every 20 m is selected except for very low speeds or when stopping, e.g. at traffic lights.

In addition to the rectified 16 bit HDR and 8 bit LDR stereo image pairs and corresponding annotations, our dataset includes disparity maps pre-computed via SGM (Gehrig et al., 2009; Hirschmüller, 2008), vehicle odometry obtained from in-vehicle sensors, outside temperature, and GPS tracks. In doing so, we ensure that all meta data that is typically leveraged by approaches for urban scene understanding is available and that multi-cue methods such as the one discussed in Chapter 3 can be run on this dataset.

#### 4.2.2 Classes and annotations

We provide fine and coarse annotations at pixel-level, extended with instance-level labels for humans and vehicles, c.f. Figure 4.1. We defined 30 visual classes for annotation, which are grouped into eight categories: flat, construction, nature, vehicle, sky, object, human, and void. Classes were selected based on their frequency, relevance from an application standpoint, practical considerations regarding the annotation effort, as well as to facilitate compatibility with existing datasets, e.g. Brostow et al., 2009; Geiger et al., 2013; Xie et al., 2016. Classes that are too rare are excluded from our benchmark, leaving 19 classes for evaluation, see Figure 4.3 for the distribution of occurrences.

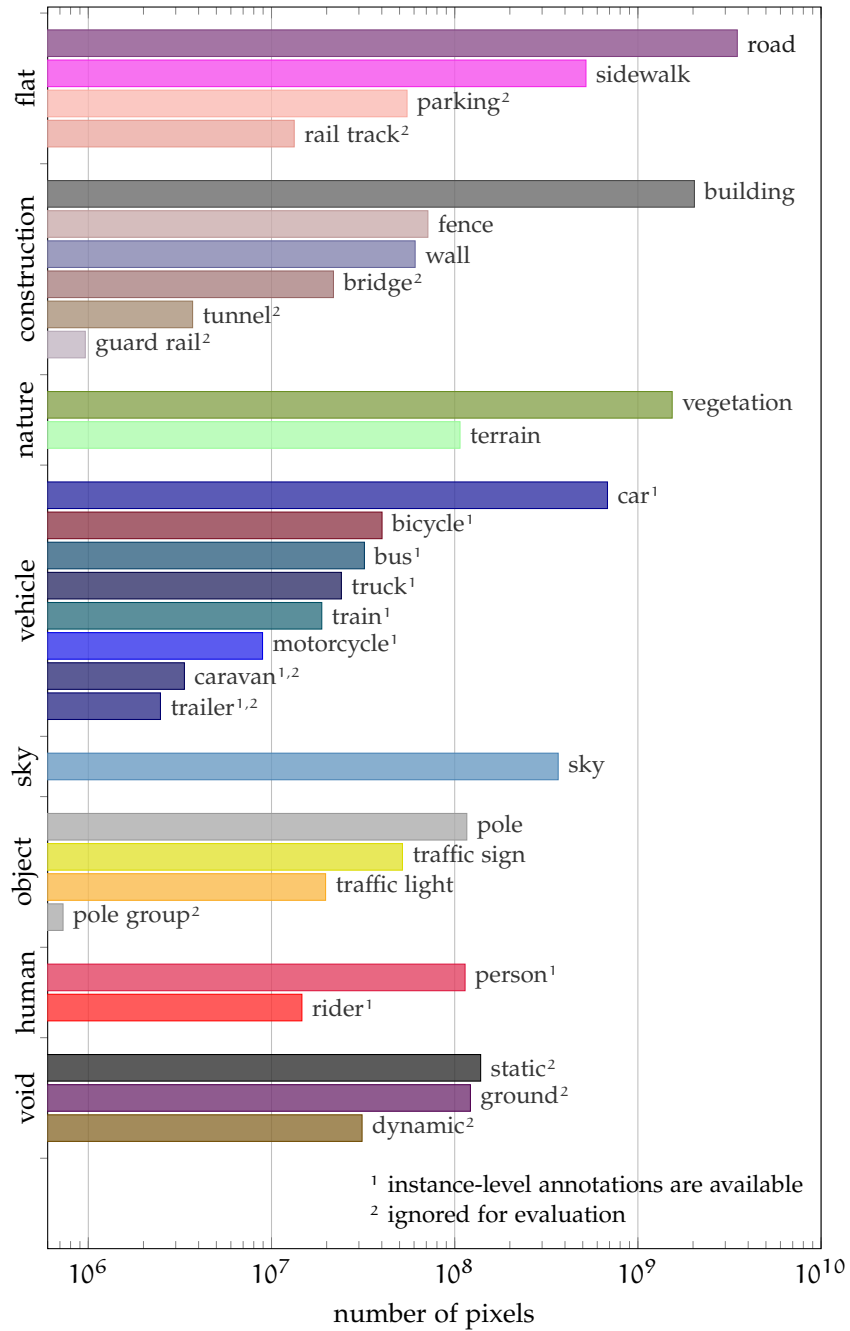


Figure 4.3: Number of finely annotated pixels (x-axis) per class and their associated categories (y-axis). The colors of the individual bars also encode the false-colors that we use to encode classes in figures within this chapter.



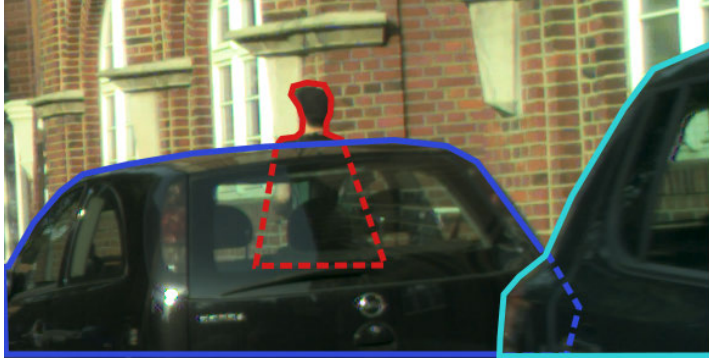


Figure 4.4: Exemplary labeling process. Distant objects are annotated first and subsequently their occluders. This ensures the boundary between these objects to be shared and consistent.

In order to achieve highest quality levels with our 5000 fine pixel-level annotations, we found several concepts and guidelines to be crucial. For all these concepts, we followed the human intuition of an urban street scene and not the possible interpretation of a concrete method running on a machine. First, we composed *clear definitions* of all the labels of interest, c.f. Appendix A.1. We included an example for each class and made sure to cover potential corner cases and to provide a precise distinction for similar classes. For example, *car* is separated from *truck* by the shape of the contour as visible in images and not by an entry in the vehicle’s paperwork, c.f. Appendix A.1. The class *van* is on purpose not included, since no clear separation from large cars and small trucks exists. Over time, we collected examples of additional corner cases and added these to the label definitions. Second, we followed a clear *label protocol*. Our annotations consist of layered polygons (à la LabelMe; Russell et al., 2008), see Figure 4.4 for an example. Annotators were asked to label the image from back to front such that no object boundary was marked more than once. In other words, distant objects are annotated first, while occluded parts are annotated with a coarser, conservative boundary (possibly larger than the actual object). Subsequently, the occluder is annotated with a polygon that lies in front of the occluded part. Thus, the boundary between these objects is shared and consistent. Each annotation thus implicitly provides a depth ordering of the objects in the scene. Given our label scheme, annotations can be easily extended to cover additional or more fine-grained classes. Third, we added rules that keep the *labeling effort in a reasonable range*. Holes in an object through which a background region can be seen are considered to be part of the object, which ensures that objects can be described via simple polygons forming simply-connected sets. We encouraged labelers to mark regions as void, where the actual object cannot be clearly identified and to use the suffix *group* (e.g. *car-group*), when the boundary between object instances with same label

cannot be determined. In doing so, these simplifications are identifiable from the annotations and corresponding image regions can be excluded during training and evaluation. Simultaneously, we keep labeling efforts in a reasonable range and avoid forcing the labelers to guess object boundaries. Fourth, we implemented a *sophisticated quality control*, where in multiple stages more experienced labelers verified results from newer ones. We supported this via a labeling tool that allowed for the labelers to explicitly mark image regions where they were unsure and that allowed for found mistakes to be fed-back to the labelers such that they could improve their skills. Further, it turned out that highest consistency is achieved when very few people perform a final quality pass. Overall, annotation and quality control required more than 1.5 h on average for a single image. Our annotation tool is published together with the dataset.

Figure 4.5 presents several examples of annotated frames from our dataset that exemplify its diversity and difficulty. All examples are taken from the *train* and *val* splits (c.f. Section 4.2.3) and were chosen by searching for the extremes in terms of the number of traffic participant instances in the scene; see Figure 4.5 for details.

For our 20 000 coarse pixel-level annotations, accuracy on object boundaries was traded off for annotation speed. We aimed to correctly annotate as many pixels as possible within a given span of less than 7 min of annotation time per image. This was achieved by labeling coarse polygons under the sole constraint that each polygon must only include pixels belonging to a single object class.

In two experiments we assessed the quality of our labeling. First, 30 images were finely annotated twice by different annotators and passed the same quality control. It turned out that 96 % of all pixels were assigned to the same label. Since our annotators were instructed to choose a *void* label if unclear (such that the region is ignored in training and evaluation), we exclude pixels having at least one *void* label and recount, yielding 98 % agreement. Second, all our fine annotations were additionally coarsely annotated such that we can enable research on densifying coarse labels. We found that 97 % of all labeled pixels in the coarse annotations were assigned the same class as in the fine annotations.

#### 4.2.3 Dataset splits

We split our densely annotated images into separate training, validation, and test sets. The coarsely annotated images serve as additional training data only. We chose not to split the data randomly, but rather in a way that ensures each split to be representative of the variability of different street scene scenarios. The underlying split criteria involve a balanced distribution of geographic location and population size of the individual cities, as well as regarding the time of year



Figure 4.5: Examples of our annotations on various images of our *train* and *val* sets. The images were selected based on criteria overlayed on each image.

when recordings took place. Specifically, each of the three split sets is comprised of data recorded with the following properties in equal shares: (i) in large, medium, and small cities; (ii) in the geographic west, center, and east; (iii) in the geographic north, center, and south; (iv) at the beginning, middle, and end of the year. Note that the data is split at the city level, i.e. a city is completely within a single split. The three bins for each characteristic, e.g. large, medium, small, are defined by splitting the cities at the tertiles of the respective characteristic. Following this scheme, we arrive at a unique split consisting of 2975 training and 500 validation images with publicly available annotations, as well as 1525 test images with annotations withheld for benchmarking purposes.

In order to assess how uniform (representative) the splits are regarding the four split characteristics, we trained a Fully Convolutional Network (FCN; Shelhamer et al., 2017) on the 500 images in our validation set. This model was then evaluated on the whole test set, as well as eight subsets thereof that reflect the extreme values of the four characteristics, c.f. Table 4.1. With the exception of the time of year, the performance is very homogeneous, varying less than 1.5 % points (often much less). Interestingly, the performance on the *end of the year* subset is 3.8 % points better than on the whole test set. We hypothesize that this is due to softer lighting conditions in the frequently cloudy fall. To verify this hypothesis, we additionally tested on images taken in low- or high-temperature conditions, c.f. Table 4.2. We find a 4.5 % point increase in low temperatures (cloudy) and a 0.9 % point decrease in warm (sunny) weather. Moreover, specifically training for either condition leads to an improvement on the respective test set, but not on the balanced set. These findings support our hypothesis and underline the importance of a dataset covering a wide range of conditions encountered in the real world in a balanced way.

#### 4.2.4 Statistical analysis

In Table 4.3 we provide an overview of Cityscapes and other related datasets. We compare the datasets in terms of the type of annotations, the meta information provided, the camera perspective, the type of scenes, and their size. The selected datasets contain real-world images and are either of large scale or focus on street scenes.

For the remainder of this section, we compare Cityscapes in greater detail with closely related datasets that are focused on autonomous driving. We analyze the datasets in terms of (i) annotation volume and density, (ii) the distribution of visual classes, and (iii) scene complexity. Regarding the first two aspects, we compare to other publicly available real-world datasets with semantic pixel-wise annotations, i.e. CamVid (Brostow et al., 2009), DUS (Scharwächter et al., 2013), and KITTI (Geiger et al., 2013).

Characteristic	Extreme	Delta
population	large	-0.2
	small	-0.9
geographic longitude	west	0.7
	east	-0.2
geographic latitude	north	-1.5
	south	0.9
time of year	early	-0.7
	late	3.8

Table 4.1: Evaluation of an FCN model trained on our validation set (500 images) and tested on different subsets of our test set. We evaluate at the first and last thirds of the four characteristics that we used to define the dataset split, please refer to the text for details. Performance is given in percentage-points as the delta to the performance of the same model evaluated on the whole test set, i.e. 58.3 %.

subset of <i>train</i>	subset of <i>test</i>		
	all	cold	warm
val	0.0	4.5	-0.9
cold	-2.4	0.9	-2.9
warm	-1.9	1.2	0.2

Table 4.2: Evaluation of three FCN models trained on our validation set (500 images) and on low- and high temperature subsets of our training set with 500 images each. We tested on the whole test set and on the low- and high temperature thirds analogously to Table 4.1 and applied the same normalization. All numbers are in percentage points.

Data	Labels	Col.	Vid.	Depth	Camera	Scene	#images	#classes
[1]	B	✓	×	×	Mixed	Mixed	150 k	1000
[2]	B	✓	×	×	Mixed	Mixed	20 k	20
[2]	C	✓	×	×	Mixed	Mixed	10 k	20
[3]	D	✓	×	×	Mixed	Mixed	20 k	400
[4]	C	✓	×	×	Mixed	Mixed	300 k	80
[5]	D, C	✓	×	Kinect	Person	Indoor	10 k	37
[6a]	B	✓	✓	Laser, Stereo	Car	Suburban	15 k	3
[6b]	D	✓	✓	Laser, Stereo	Car	Suburban	700	8
[7]	D	✓	✓	×	Car	Urban	701	32
[8]	D	✓	✓	Stereo, Manual	Car	Urban	70	7
[9]	D	×	✓	Stereo	Car	Urban	500	5
[10]	B	✓	✓	×	Car	Urban	250 k	1
[11]	D	✓	×	×	Person	Urban	200	2
[12]	C	✓	×	Stereo	Car	Facades	86	13
[13]	D	✓	×	3D mesh	Person	Urban	428	8
[14]	D	✓	✓	Laser	Car	Suburban	400 k	27
Ours	D	✓	✓	Stereo	Car	Urban	5 k	30
Ours	C	✓	✓	Stereo	Car	Urban	20 k	30

- [1] ImageNet: Russakovsky et al., 2015  
[2] PASCAL VOC: Everingham et al., 2014  
[3] PASCAL-Context: Mottaghi et al., 2014  
[4] Microsoft COCO: T.-Y. Lin et al., 2014  
[5] SUN RGB-D: Song et al., 2015  
[6a] KITTI detection: Geiger et al., 2013  
[6b] KITTI segmentation: Geiger et al., 2013, including the annotations of 3<sup>rd</sup>-party groups: Güney and Geiger, 2015; H. He and Upcroft, 2013; Kundu et al., 2014; Ladický et al., 2014; Ros et al., 2015; Sengupta et al., 2013; C. Xu et al., 2013; R. Zhang et al., 2015  
[7] CamVid: Brostow et al., 2009  
[8] Leuven: Leibe et al., 2007  
[9] DUS: Scharwächter et al., 2013  
[10] Caltech: Dollár et al., 2012  
[11] Geosemantic segmentation: Ardeshir et al., 2015  
[12] Automatic Dense Visual Semantic Mapping: Sengupta et al., 2012  
[13] RueMonge: Riemenschneider et al., 2014  
[14] 3D to 2D Label Transfer: Xie et al., 2016

Table 4.3: Comparison to related datasets. We list the type of labels provided, i.e. object bounding boxes (B), dense pixel-level semantic labels (D), and coarse labels (C) that do not aim to label the whole image. Further, we mark if color, video, and depth information are available. We list the camera perspective, the scene type, the number of images, and the number of semantic classes.



	#pixels [ $10^9$ ]	annot. density [%]
Ours (fine)	9.43	<b>97.1</b>
Ours (coarse)	<b>26.0</b>	67.5
CamVid	0.62	96.2
DUS	0.14	63.0
KITTI	0.23	88.9

Table 4.4: Absolute number and density of annotated pixels for Cityscapes, DUS, KITTI, and CamVid (upscaled to  $1280 \times 720$  pixels to maintain the original aspect ratio).

CamVid consists of ten minutes of video footage with pixel-wise annotations for over 700 frames. DUS consists of a video sequence of 5000 images from which 500 have been annotated. KITTI addresses several different tasks including semantic labeling and object detection. As no official pixel-wise annotations exist for KITTI, several independent research groups have annotated approximately 700 frames (Güney and Geiger, 2015; H. He and Upcroft, 2013; Kundu et al., 2014; Ladický et al., 2014; Ros et al., 2015; Sengupta et al., 2013; C. Xu et al., 2013; R. Zhang et al., 2015). We map those labels to our high-level categories and analyze this consolidated set. In comparison, Cityscapes provides significantly more annotated images, i.e. 5000 fine and 20 000 coarse annotations. Moreover, the annotation quality and richness is notably better. As Cityscapes provides recordings from 50 different cities, it also covers a significantly larger area than previous datasets that contain images from a single city only, e.g. Cambridge (CamVid), Heidelberg (DUS), and Karlsruhe (KITTI). In terms of absolute and relative numbers of semantically annotated pixels (training, validation, and test data), Cityscapes compares favorably to CamVid, DUS, and KITTI with up to two orders of magnitude more annotated pixels, c.f. Table 4.4. The majority of all annotated pixels in Cityscapes belong to the coarse annotations, providing many individual (but correlated) training samples, but missing information close to object boundaries.

Figures 4.3 and 4.6 compare the distribution of annotations across individual classes and their associated higher-level categories. Notable differences stem from the inherently different configurations of the datasets. Cityscapes involves dense inner-city traffic with wide roads and large intersections, whereas KITTI is composed of less busy suburban traffic scenes. As a result, KITTI exhibits significantly fewer *flat* ground structures, fewer *humans*, and more *nature*. In terms of overall composition, DUS and CamVid seem more aligned with our dataset. Exceptions are an abundance of *sky* pixels in CamVid due to cameras with a comparably large vertical field-of-view and the absence of certain categories in DUS, i.e. *nature* and *object*.

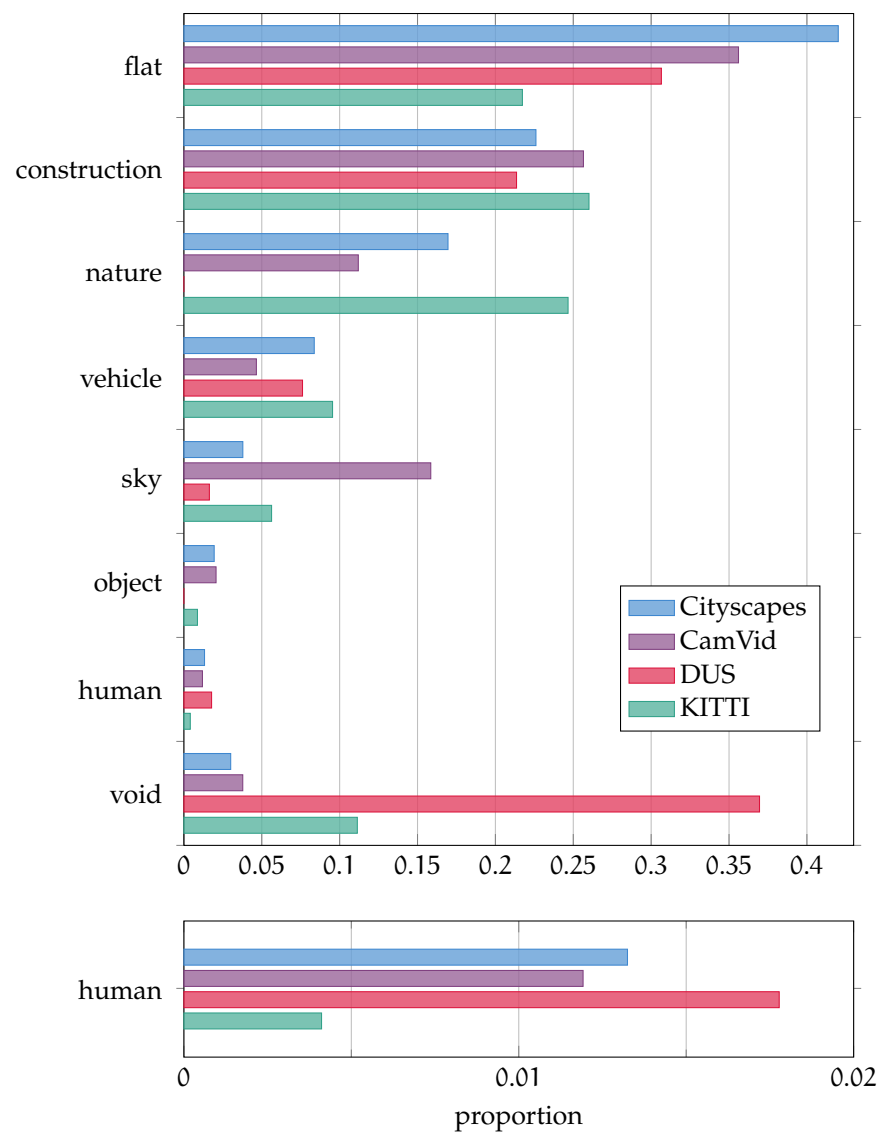


Figure 4.6: Proportion of annotated pixels (x-axis) per category (y-axis) for Cityscapes, CamVid, DUS, and KITTI.



	#humans [10 <sup>3</sup> ]	#vehicles [10 <sup>3</sup> ]	#humans per image	#vehicles per image
Ours (fine)	24.4	<b>41.0</b>	<b>7.0</b>	<b>11.8</b>
<a href="#">KITTI</a>	6.1	30.3	0.8	4.1
Caltech	<b>192<sup>a</sup></b>	-	1.5	-

<sup>a</sup>via interpolation

Table 4.5: Absolute and average number of instances for Cityscapes, [KITTI](#), and Caltech on the respective training and validation datasets.

Finally, we assess scene complexity, where density and scale of traffic participants (humans and vehicles) serve as proxy measures. Out of the previously discussed datasets, only Cityscapes and [KITTI](#) provide instance-level annotations for humans and vehicles. We additionally compare to the Caltech Pedestrian Dataset (Dollár et al., 2012), which only contains annotations for humans, but none for vehicles. Furthermore, [KITTI](#) and Caltech only provide instance-level annotations in terms of axis-aligned bounding boxes. We use the respective training and validation splits for our analysis, since test set annotations are not publicly available for all datasets. In absolute terms, Cityscapes contains significantly more object instance annotations than [KITTI](#), see Table 4.5. Being a specialized benchmark, the Caltech dataset provides the most annotations for humans by a margin. The major share of those labels was obtained, however, by interpolation between a sparse set of manual annotations resulting in significantly degraded label quality. The relative statistics emphasize the much higher complexity of Cityscapes, as the average numbers of object instances per image notably exceed those of [KITTI](#) and Caltech. We extend our analysis to Microsoft [COCO](#) (T.-Y. Lin et al., 2014) and [PASCAL VOC](#) (Everingham et al., 2014) that also contain street scenes while not being specific for them. We analyze the frequency of scenes with a certain number of traffic participant instances, see Figure 4.7. We find our dataset to cover a greater variety of scene complexity and to have a higher portion of highly complex scenes than previous datasets. Using stereo data, we analyze the distribution of vehicle distances to the camera. From Figure 4.8 we observe that in comparison to [KITTI](#), Cityscapes covers a larger distance range. We attribute this to both our higher-resolution imagery and the careful annotation procedure. As a consequence, algorithms need to take a larger range of scales and object sizes into account to score well in our benchmark. Overall, we find that our dataset covers a greater range of scene complexity and has a larger portion of highly complex scenes than the datasets compared with.

Recently, researchers started to use alternatives to standard image labeling to create large-scale datasets. Xie et al., 2016 announced

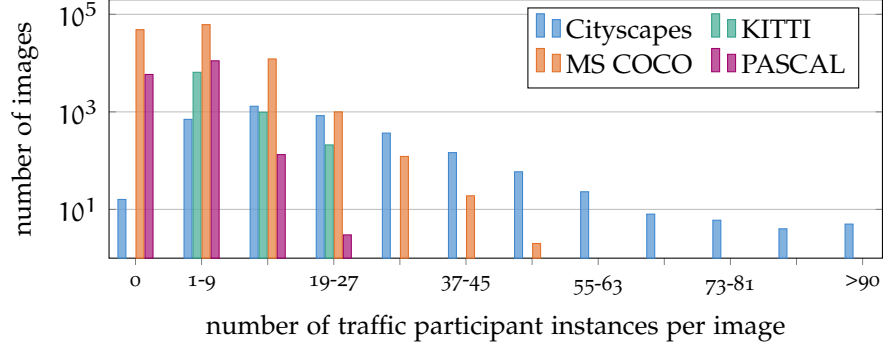


Figure 4.7: Dataset statistics regarding scene complexity. Only Microsoft COCO and Cityscapes provide instance segmentation masks.

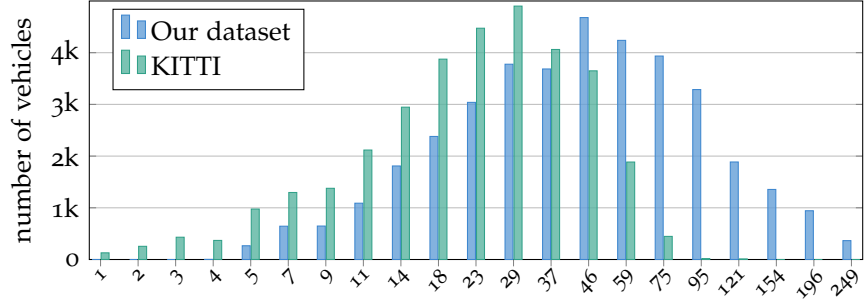


Figure 4.8: Histogram of object distances in meters for class *vehicle*.

a new semantic scene labeling dataset for suburban traffic scenes. It provides temporally consistent 3D semantic instance annotations with 2D annotations obtained through back-projection. We consider our efforts to be complementary given the differences in the way that semantic annotations are obtained, and in the type of scenes considered, i.e. suburban vs. inner-city traffic. To maximize synergies between both datasets, a common label definition that allows for cross-dataset evaluation has been mutually agreed upon and implemented. Another novel trend is to extract datasets from rendered scenes, either via simulators (Ros et al., 2016) or via video games (Richter et al., 2016; Shafaei et al., 2016). For the future, it will be exciting to combine such datasets, to develop methods for cross-dataset generalization, and to find the optimal trade-off between manual labeling efforts and automated dataset generation via rendering. We believe that Cityscapes will aid such investigations by providing a real-world representative benchmark suite.

#### 4.3 PIXEL-LEVEL SEMANTIC LABELING

Besides the dataset itself, Cityscapes consists of a benchmark suite to evaluate and fairly compare different approaches, c.f. Section 4.5. For this benchmark suite, we focus on two major tasks for autonomous

driving. The first task involves predicting a per-pixel semantic labeling of the image without considering higher-level object instance or boundary information and is referred to as *pixel-level semantic labeling*.

#### 4.3.1 Metrics

To assess labeling performance, we rely on a standard and a novel metric. The first is the standard Jaccard Index, commonly known as the [PASCAL VOC](#) Intersection-over-Union ([IoU](#)) metric (Everingham et al., 2014) and defined as

$$\text{IoU} = \frac{\sum \text{TP}}{\sum \text{TP} + \sum \text{FP} + \sum \text{FN}}, \quad (4.1)$$

where  $\sum \text{TP}$ ,  $\sum \text{FP}$ , and  $\sum \text{FN}$  are the numbers of true positive, false positive, and false negative pixels, respectively, determined over the whole test set. Owing to the two semantic granularities, i.e. classes and categories, we report two separate mean performance scores:  $\text{IoU}_{\text{class}}$  and  $\text{IoU}_{\text{category}}$ . In either case, pixels labeled as void do not contribute to the score. Note that this metric was previously introduced in greater detail, c.f. Section 3.8.3.

It is well-known that the global [IoU](#) measure is biased toward object instances that cover a large image area. In street scenes with their strong scale variation this can be problematic. Specifically for traffic participants, which are the key classes in our scenario, we aim to evaluate how well the individual instances in the scene are represented in the labeling. To address this, we introduced an additional object-centric evaluation in Section 3.8.3. For the Cityscapes pixel-level metrics, we would like to follow this idea, but with a slightly altered metric. The object-centric evaluation from Section 3.8.3 has two drawbacks: it is rather complex and hard to understand and is computationally expensive to evaluate since for each pixel in the image the closest candidate instance must be searched, c.f. Algorithm 3.2. Especially the latter renders this metric infeasible for an evaluation server on such a large-scale dataset with 2MP images. To overcome these limitations, we propose an instance-level intersection-over-union metric defined as

$$\text{iIoU} = \frac{\sum \text{iTP}}{\sum \text{iTP} + \sum \text{FP} + \sum \text{iFN}}. \quad (4.2)$$

Here,  $\sum \text{iTP}$ , and  $\sum \text{iFN}$  denote weighted counts of true positive and false negative pixels, respectively. In contrast to the standard [IoU](#) measure, the contribution of each pixel is weighted by the ratio of the class' average instance size to the size of the respective ground truth instance, where sizes are measured as the number of pixels assigned to the instances. As before,  $\sum \text{FP}$  is the number of false positive pixels. It is important to note here that unlike the instance-level task in Section 4.4, we assume that the methods only yield a

standard per-pixel semantic class labeling as output. Therefore, the false positive pixels are not associated with any instance and thus do not require normalization. The final scores,  $iIoU_{\text{class}}$  and  $iIoU_{\text{category}}$ , are obtained as the means for the two semantic granularities, while only classes with instance annotations are included, i.e. vehicles and humans.

#### 4.3.2 Control experiments

We conduct several control experiments to put our baseline results below into perspective. Average results with respect to all of our metrics introduced in Section 4.3.1 are listed in Table 4.6, but for simplicity we focus in the following discussion on the main score  $IoU_{\text{class}}$ . Performance numbers for individual classes are provided in Tables 4.7 and 4.8 and qualitative results of the control experiments can be found in Figures 4.9 and 4.10.

First, we count the relative frequency of every class label at each pixel location of the fine (coarse) training annotations. We use the most frequent label at each pixel as a constant prediction irrespective of the test image and call this the Static Predictor Fine (SPF) and Static Predictor Coarse (SPC), respectively. This control experiment results in roughly 10 %  $IoU_{\text{class}}$ , as shown in Table 4.6. These low scores emphasize the high diversity of our data. SPF and SPC having similar performance indicates the value of our additional coarse annotations. Even if the ground truth segments are re-classified using the most frequent training label (SPF or SPC) within each segment mask, the performance does not notably increase.

Secondly, we re-classify each ground truth segment using FCN-8s (Shelhamer et al., 2017), c.f. Section 4.3.3. We compute the average scores within each segment and assign the maximizing label. The performance is significantly better than the static predictors but still far from 100 %. We conclude that it is necessary to optimize both classification and segmentation quality at the same time.

Thirdly, we evaluate the performance of subsampled ground truth annotations as predictors. Subsampling was done by majority voting of neighboring pixels, followed by resampling back to full resolution. This yields an upper bound on the performance at a fixed output resolution and is particularly relevant for deep learning approaches that often apply downscaling due to constraints on time, memory, or the network architecture itself. Note that while subsampling by a factor of 2 hardly affects the  $IoU$  score, it clearly decreases the  $iIoU$  score given its comparatively large impact on small, but nevertheless important objects. This underlines the importance of the separate instance-normalized evaluation. The downsampling factors of 8, 16, and 32 correspond to the strides of the FCN baseline model presented in Section 4.3.3. Downsampling by a factor of 128 is the smallest

	Average over Metric [%]	Classes		Categories	
		IoU	iIoU	IoU	iIoU
Static Predictor Fine (SPF)		10.1	4.7	26.3	19.9
Static Predictor Coarse (SPC)		10.3	5.0	27.5	21.7
GT <sup>a</sup> segmentation with SPF		10.1	6.3	26.5	25.0
GT segmentation with SPC		10.9	6.3	29.6	27.0
GT segmentation with FCN <sup>b</sup>		79.4	52.6	93.3	80.9
GT subsampled by 2		97.2	92.6	97.6	93.3
GT subsampled by 4		95.2	90.4	96.0	91.2
GT subsampled by 8		90.7	82.8	92.1	83.9
GT subsampled by 16		84.6	70.8	87.4	72.9
GT subsampled by 32		75.4	53.7	80.2	58.1
GT subsampled by 64		63.8	35.1	71.0	39.6
GT subsampled by 128		50.6	21.1	60.6	29.9
nearest training neighbor		21.3	5.9	39.7	18.6

<sup>a</sup>Ground Truth<sup>b</sup>Long et al., 2015

Table 4.6: Quantitative results of control experiments for pixel-level semantic labeling using the metrics presented in Section 4.3.1.

(power of 2) downsampling for which all images have a distinct labeling.

Lastly, we employ 128-times subsampled annotations and retrieve the nearest training annotation in terms of the Hamming distance. The full resolution version of this training annotation is then used as prediction, resulting in 21 % IoU<sub>class</sub>. While outperforming the static predictions, the poor result demonstrates the high variability of our dataset and its demand for approaches that generalize well.

#### 4.3.3 Baselines

Our own baseline experiments rely on Fully Convolutional Networks (FCNs), as they are central to most state-of-the-art methods, e.g. L.-C. Chen et al., 2016; G. Lin et al., 2016; Schwing and Urtasun, 2015; Shelhamer et al., 2017; S. Zheng et al., 2015. We adopted a VGG-16 network (Simonyan and Zisserman, 2015) and start with a model pretrained on ImageNet (Russakovsky et al., 2015), i.e. the underlying CNN was initialized with ImageNet VGG weights. We additionally investigated first pretraining on PASCAL-Context (Mottaghi et al., 2014), but found this to not influence performance given a sufficiently large number of

	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
Static Predictor Fine (SPF)	22.1	0.0	0.0	23.4	0.0	0.0	0.0	0.0	0.0
Static Predictor Coarse (SPC)	24.3	0.0	0.0	26.2	0.0	0.0	0.0	0.0	0.0
GT <sup>a</sup> segmentation with SPF	17.9	0.0	0.0	32.9	0.0	0.0	0.0	0.0	0.0
GT segmentation with SPC	29.2	0.0	0.0	34.1	0.0	0.0	0.0	0.0	0.0
GT segmentation with FCN <sup>b</sup>	99.1	90.6	69.2	98.0	59.0	66.9	71.6	66.8	85.8
GT subsampled by 2	98.3	96.5	95.7	98.9	98.9	99.1	98.9	96.5	95.8
GT subsampled by 4	97.9	94.1	92.5	98.2	98.1	98.5	98.1	94.1	93.0
GT subsampled by 8	94.5	88.9	86.1	96.2	95.9	96.7	96.1	88.7	86.8
GT subsampled by 16	93.1	81.0	76.0	93.5	93.0	94.4	93.4	80.8	78.0
GT subsampled by 32	88.7	69.4	62.3	88.0	87.4	89.8	88.5	68.6	65.6
GT subsampled by 64	81.6	55.1	46.4	78.8	78.9	82.4	80.2	54.2	50.7
GT subsampled by 128	69.9	41.1	31.5	67.3	66.3	70.1	68.3	36.0	33.3
nearest training neighbor	36.5	4.0	0.4	42.0	9.7	18.3	12.9	0.3	1.7

<sup>a</sup>Ground Truth<sup>b</sup>Long et al., 2015

Table 4.7: Detailed results of our control experiments for the pixel-level semantic labeling task in terms of the IoU score on the class level. We list results for the classes in the categories *sky*, *human*, and *vehicle*. The remaining categories are listed in Table 4.8. All numbers are given in percent.

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain
Static Predictor Fine (SPF)	80.0	13.2	40.3	0.0	0.0	0.0	0.0	0.0	12.5	0.0
Static Predictor Coarse (SPC)	80.1	9.5	39.5	0.0	0.0	0.0	0.0	0.0	16.4	0.0
GT <sup>a</sup> segmentation with SPF	80.8	11.1	44.5	0.0	0.0	0.0	0.0	0.0	4.2	0.0
GT segmentation with SPC	79.6	5.1	46.6	0.0	0.0	0.0	0.0	0.0	11.8	0.0
GT segmentation with FCN <sup>b</sup>	99.3	91.9	94.8	44.9	62.0	66.1	81.2	84.3	96.5	80.1
GT subsampled by 2	99.6	98.1	98.6	97.8	97.4	90.4	94.1	95.2	98.7	97.6
GT subsampled by 4	99.4	96.8	98.0	96.1	95.5	83.1	89.7	91.6	98.0	96.0
GT subsampled by 8	98.6	93.4	95.4	92.3	91.1	69.5	80.9	84.2	95.5	92.1
GT subsampled by 16	97.8	88.8	93.1	86.9	84.9	50.9	68.4	73.0	93.4	86.5
GT subsampled by 32	96.0	80.9	88.7	77.6	75.2	30.9	51.6	56.8	89.2	77.3
GT subsampled by 64	92.1	69.6	83.0	65.5	61.0	14.8	32.1	37.6	83.3	65.2
GT subsampled by 128	86.2	55.0	75.2	51.3	45.9	5.7	13.6	17.9	75.2	51.6
nearest training neighbor	85.3	35.6	56.7	15.6	6.2	1.3	0.5	1.0	54.2	23.3

<sup>a</sup>Ground Truth<sup>b</sup>Long et al., 2015

Table 4.8: Detailed results of our control experiments for the pixel-level semantic labeling task in terms of the IoU score on the class level. We list results for the classes in the categories *flat*, *construction*, *object*, and *nature*. The remaining categories are listed in Table 4.7. All numbers are given in percent.

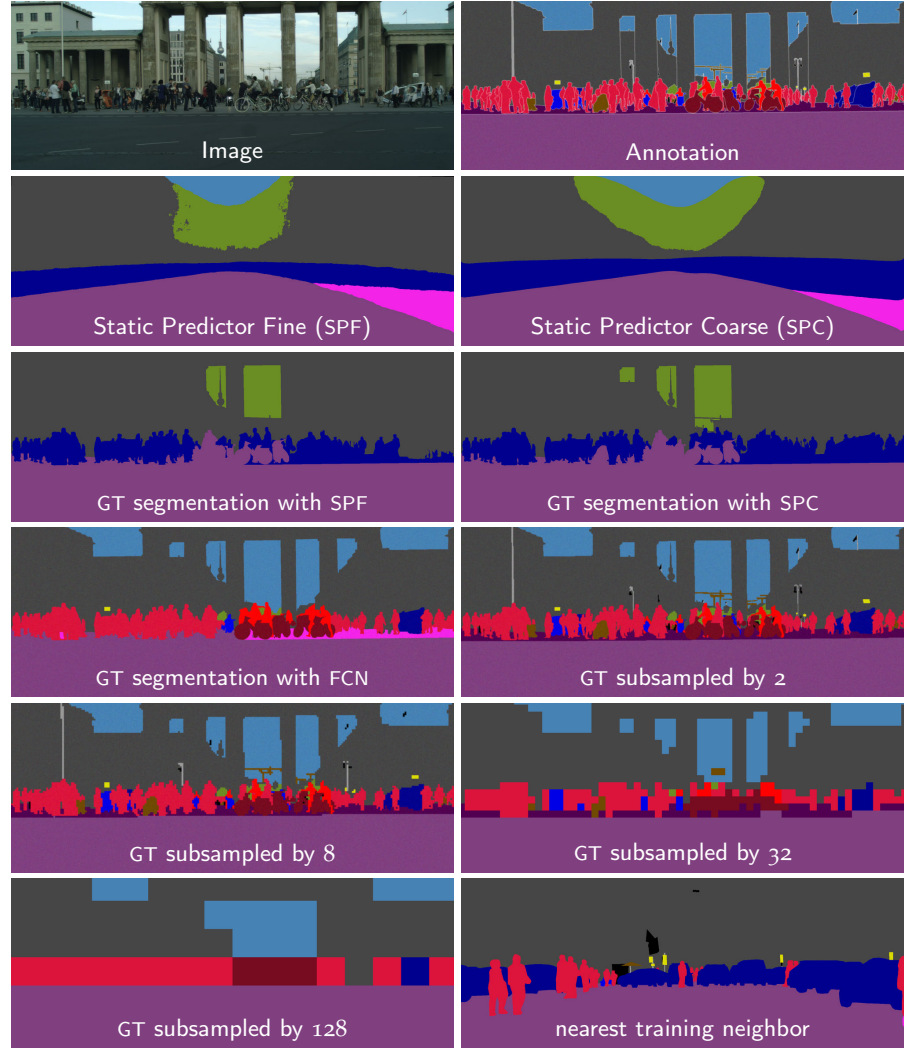


Figure 4.9: Exemplary output of our control experiments for the pixel-level semantic labeling task. The image is part of our *test* set and has both, the largest number of instances and persons.



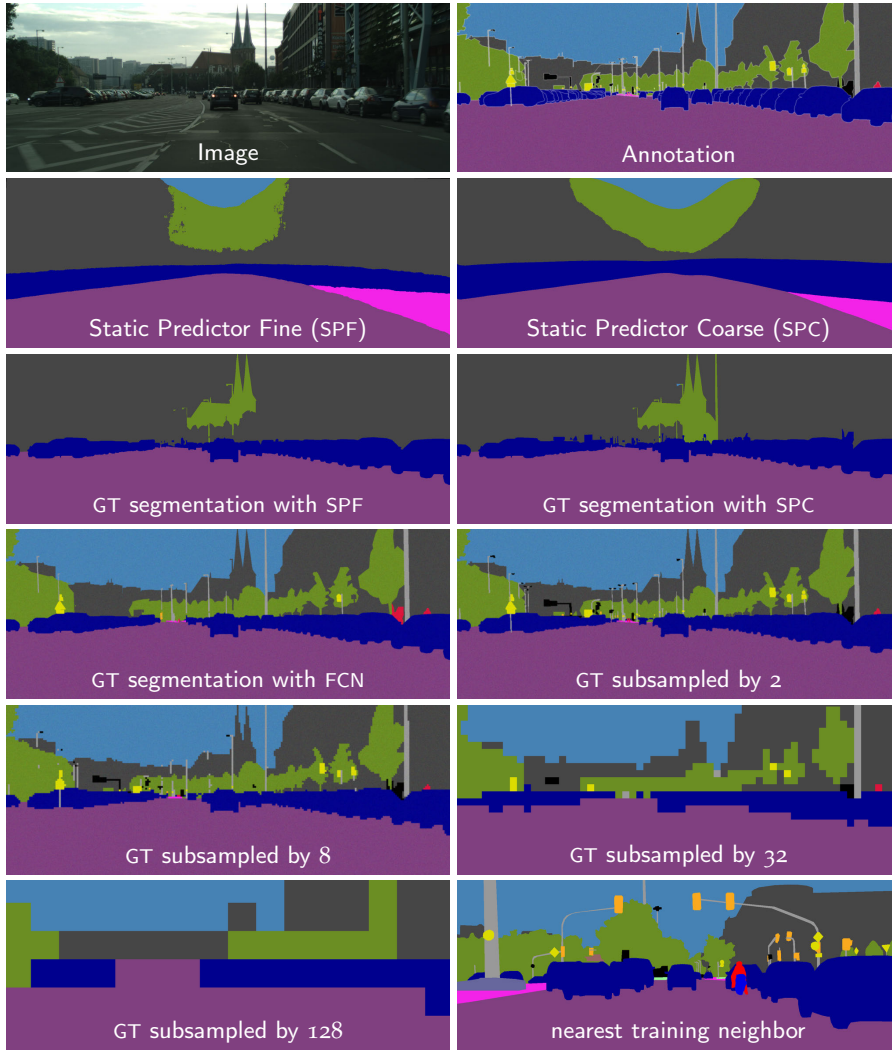


Figure 4.10: Exemplary output of our control experiments for the pixel-level semantic labeling task. The image is part of our *test* set and has the largest number of car instances.

training iterations. For training and testing, we rely on single frame, monocular LDR images only.

After initialization, the FCN is finetuned on Cityscapes using the respective portions listed in Table 4.9. For finetuning, we utilized the training scripts that were published in conjunction with the conference paper by Long et al., 2015, which is a predecessor of the journal article (Shelhamer et al., 2017). We base on the models for the PASCAL-context dataset and use the Caffe framework (Jia et al., 2014) for training. The only change that we made was to modify the learning rate to match our image resolution under an unnormalized loss. In doing so, we ensured to stay as close as possible to the original implementation. Since VGG-16 training on 2MP images exceeds even the largest GPU memory available, we split each image into two halves with an overlap that is sufficiently large considering the network’s receptive field. As proposed by Long et al., 2015, we use three-stage training with subsequently smaller strides, i.e. first FCN-32s, then FCN-16s, and then FCN-8s, always initializing with the parameters from the previous stage. Note that the numbers denote the stride of the finest heatmap. We add a 4<sup>th</sup> stage for which we reduce the learning rate by a factor of 10. Each stage is trained until convergence on the validation set; pixels with *void* ground truth are ignored such that they do not induce any gradient. Eventually, we retrain on *train* and *val* together with the same number of epochs, yielding 243 250, 69 500, 62 550, and 5950 iterations for stages 1 through 4. Note that each iteration corresponds to half of an image (see above). For the variant with factor 2 downsampling, no image splitting is necessary, yielding 80 325, 68 425, 35 700, and 5950 iterations in the respective stages. The variant only trained on *val* (full resolution) uses *train* for validation, leading to 130 000, 35 700, 47 600, and 0 iterations in the 4 stages. Our last FCN variant is trained using the coarse annotations only, with 386 750, 113 050, 35 700, and 0 iterations in the respective stage; pixels with *void* ground truth are ignored here as well.

Quantitative results of our baseline experiments can be found in Tables 4.9 to 4.11 as well as qualitative examples in Figures 4.11 and 4.12. From studying these results, we can draw some early conclusions: (1) Training the FCN models for finer heatmaps yields better accuracy. Note that the significant improvement from FCN-16s to FCN-8s stands in contrast to the observations in Shelhamer et al., 2017, confirming the high quality of annotations in Cityscapes such that the evaluation is sensitive to finer predictions at the object boundaries, c.f. Figures 4.11 and 4.12. (2) The amount of downscaling applied during training and testing has a strong negative influence on performance (c.f. FCN-8s vs. FCN-8s at half resolution). We attribute this to the large scale variation present in our dataset, c.f. Figure 4.8. (3) Training FCN-8s with 500 densely annotated images (750 h of annotation) yields comparable IoU performance to a model trained on 20 000

	train	val	coarse	sub	Classes		Categories	
					IoU	iIoU	IoU	iIoU
FCN-32s	✓	✓			61.3	38.2	82.2	65.4
FCN-16s	✓	✓			64.3	41.1	84.5	69.2
FCN-8s	✓	✓			65.3	41.7	85.7	70.1
FCN-8s sub	✓	✓		2	61.9	33.6	81.6	60.9
FCN-8s val		✓			58.3	37.4	83.4	67.2
FCN-8s coarse			✓		58.0	31.8	78.2	58.4

Table 4.9: Quantitative results of baselines for semantic labeling using the metrics presented in Section 4.3.1. We show results for different Fully Convolutional Network (FCN) models (Long et al., 2015) trained on different sets of training data. All numbers are given in percent and we indicate the used training data for each method, i.e. *train fine*, *val fine*, *coarse extra* as well as a potential downscaling factor (sub) of the input image.

weakly annotated images (1300h of annotation). However, in both cases the performance is significantly lower than FCN-8s trained on all 3475 densely annotated images. Many fine labels are thus important for training standard methods as well as for testing, but the performance using coarse annotations only does not collapse and presents a viable option. (4) The iIoU metric treats instances of any size equally and is therefore more sensitive to errors in predicting small objects compared to the IoU. Since the coarse annotations do not include small or distant instances, their iIoU performance is worse.

#### 4.3.4 Pixel-level benchmark

A major goal of the Cityscapes benchmark suite is to enable a fair comparison of the performances of different methods for semantic urban scene understanding. To achieve this goal, we followed two concepts. First, prior to a public release of the dataset, we asked selected groups that proposed state-of-the-art semantic labeling approaches in the past to optimize and run their methods on our dataset, such that we could evaluate their predictions on our test set. Second, concurrently with the official release, we launched an evaluation server, where researchers can upload their predictions on the test set enabling a fully automated evaluation, c.f. Section 4.5. In both cases, it is ensured that the participants do not have access to the test set and in turn overfitting is prevented.

For the scope of this dissertation, we provide and analyze a snapshot of the leaderboard for pixel-level semantic labeling as of April 15<sup>th</sup>, 2017. We only consider those submissions that are publicly

	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
FCN-32s	91.6	71.1	46.7	91.0	33.3	46.6	43.8	48.2	59.1
FCN-16s	92.9	75.4	50.5	91.9	35.3	49.1	45.9	50.7	65.2
FCN-8s	93.9	77.1	51.4	92.6	35.3	48.6	46.5	51.6	66.8
FCN-8s sub	92.5	69.5	46.0	90.8	41.9	52.9	50.1	46.5	58.4
FCN-8s val	92.9	73.3	42.7	89.9	22.8	39.2	29.6	42.5	63.1
FCN-8s coarse	90.2	59.6	37.2	86.1	35.4	53.1	39.7	42.6	52.6

Table 4.10: Detailed results of our baseline experiments for the pixel-level semantic labeling task in terms of the **IoU** score on the class level. We list results for the classes in the categories *sky*, *human*, and *vehicle*. The remaining categories are listed in Table 4.11. All numbers are given in percent.

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain
FCN-32s	97.1	76.0	87.6	33.1	36.3	35.2	53.2	58.1	89.5	66.7
FCN-16s	97.3	77.6	88.7	34.7	44.0	43.0	57.7	62.0	90.9	68.6
FCN-8s	97.4	78.4	89.2	34.9	44.2	47.4	60.1	65.0	91.4	69.3
FCN-8s sub	97.0	75.4	87.3	37.4	39.0	35.1	47.7	53.3	89.3	66.1
FCN-8s val	95.9	69.7	86.9	23.1	32.6	44.3	52.1	56.8	90.2	60.9
FCN-8s coarse	95.3	67.7	84.6	35.9	41.0	36.0	44.9	52.7	86.6	60.2

Table 4.11: Detailed results of our baseline experiments for the pixel-level semantic labeling task in terms of the **IoU** score on the class level. We list results for the classes in the categories *flat*, *construction*, *object*, and *nature*. The remaining categories are listed in Table 4.10. All numbers are given in percent.

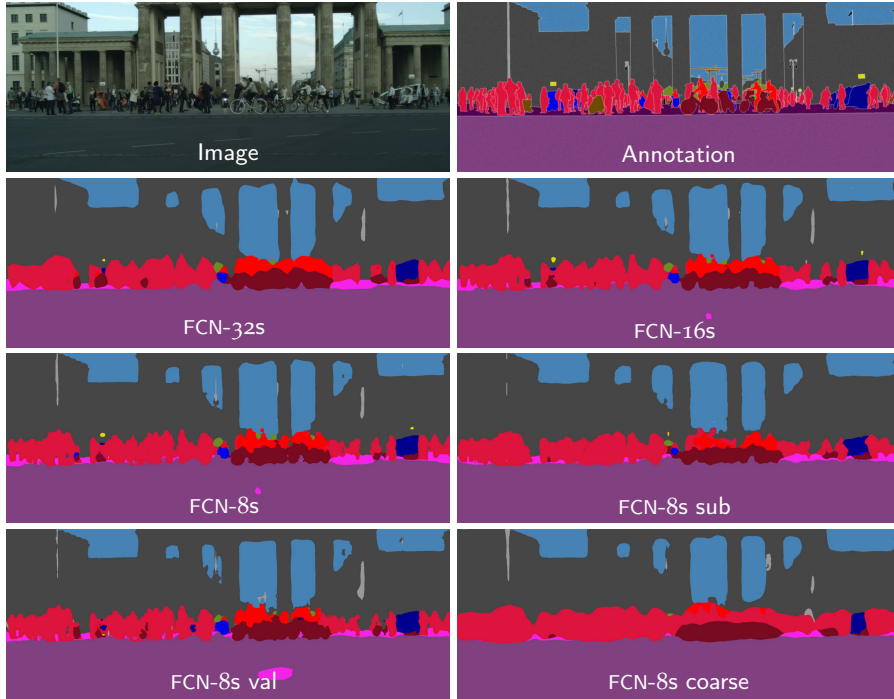


Figure 4.11: Exemplary output of our baseline experiments for the pixel-level semantic labeling task. The image is part of our *test* set and has both, the largest number of instances and persons.

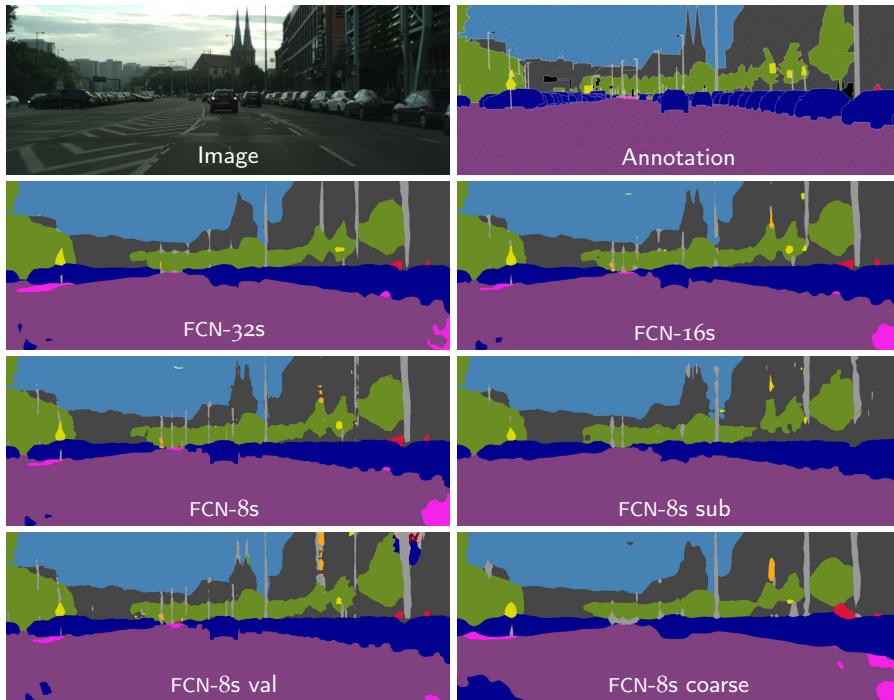


Figure 4.12: Exemplary output of our baseline experiments for the pixel-level semantic labeling task. The image is part of our *test* set and has the largest number of car instances.

listed, yielding 48 submissions. Next, we filter out those entries that are anonymous, i.e. do not refer to a publication, which holds for approximately 44 % of the public submissions. In cases where there are multiple variants of the same method listed, we only keep the best performing, leaving 20 submissions in our snapshot. We then selected the ten best performing approaches for our study. In addition, matching the research goals of this dissertation, we include methods that leverage multiple input modalities, and the two best performing methods that report a run time of less than 500 ms. Note that we also included our real time model that we will present in Section 4.3.5. An overview of the approaches is given in Table 4.12 and all methods were already briefly presented in Chapter 2. The quantitative performance numbers of the methods are evaluated for each class and reported in Tables 4.14 and 4.15, while the average performance is provided in Table 4.13. Qualitative results of the top eight as well as the additionally selected methods can be found in Figures 4.13 and 4.14. The best performing model is *ResNet-38* by Z. Wu et al., 2016 with an average  $\text{IoU}_{\text{class}}$  score of 80.6 %. Z. Wu et al., 2016 propose to widen neural networks instead of deepening them and combine such a CNN based on *ResNet* (K. He et al., 2016) with the *DeepLab* framework (L.-C. Chen et al., 2016). Impressively, considering class-level accuracy, this method outperforms our oracle experiment, where we used an *FCN* to classify ideal segments, c.f. Table 4.6.

We start our discussion with a high-level analysis of the selected methods. Following the current trend in the computer vision community, all methods rely on CNNs for pixel classification. To this end, most authors base on network architectures originally designed for image classification such as *VGG* (Simonyan and Zisserman, 2015) or *ResNet* (K. He et al., 2016). These networks are then slightly modified to allow for a dense pixel-level classification, but the changes are small enough such that using a pretrained ImageNet (Russakovsky et al., 2015) classification model for initialization remains possible. After initialization, the network is trained on the target data, i.e. Cityscapes, a procedure commonly referred to as *finetuning*. While such an approach limits the degrees of freedom, the network benefits from generic features that were learned for large-scale image classification, training converges faster, and the performance is typically increased due to a network that generalizes better. Even though Cityscapes is a large-scale dataset itself, it is biased towards urban street scenes and additional generic data helps to increase recognition performance. Note that we also adopted this concept for our baseline experiments in Section 4.3.3. Within the analyzed methods, the only exceptions are *FRRN* (Pohlen et al., 2017) and *ENet* (Paszke et al., 2016) where the networks are specifically designed for high-resolution pixel classification and efficiency, respectively. Both networks are only trained on Cityscapes and are initialized randomly.



Name	Reference	depth	coarse	base net	CRF	mst	sub	rt
ResNet-38	Z. Wu et al., 2016		✓	A		✓		
PSPNet	Zhao et al., 2017		✓	R		✓		
TuSimple	Wang et al., 2017		✓	R				
RefineNet	G. Lin et al., 2017			R		✓		
LRR-4x	Ghiasi and Fowlkes, 2016		✓	V				
FRRN	Pohlen et al., 2017						2	
Adelaide Context	G. Lin et al., 2016			V	✓			
Deep Layer Cascade (LC)	Xiaoxiao Li et al., 2017			I				
DeepLab v2 CRF	L.-C. Chen et al., 2016			R	✓			
Dilation 10	Yu and Koltun, 2016			V				4k
Scale invariant CNN	Kreso et al., 2016	✓		V	✓			
SQ	Treml et al., 2016			S				60
ENet	Paszke et al., 2016						2	13
Ours	Section 4.3.5		✓	G				44

Pretrained networks for initialization on ImageNet (Russakovsky et al., 2015):

R	ResNet-101	K. He et al., 2016
V	VGG-16	Simonyan and Zisserman, 2015
I	Inception ResNet v2	Szegedy et al., 2016a
S	SqueezeNet v1.1	Iandola et al., 2016
G	GoogLeNet v1	Szegedy et al., 2015

on ImageNet (Russakovsky et al., 2015) and Places 365 (Zhou et al., 2016):

A	A2, 2 conv	Z. Wu et al., 2016
---	------------	--------------------

Table 4.12: Overview of included methods in the pixel-level semantic labeling benchmark. We drew a snapshot of the public leaderboard on April 15, 2017 and include the top ten methods, the two fastest, as well as those using multi-cue input data. For each method, we provide a name and a reference, we indicate if depth meta data was used (other meta data was never used), if coarse annotations were used during training, if a pretrained model was used for network initialization (base net), if a CRF is used as post-processing, if multi-scale testing (mst) was applied to increase test performance, a potential downscaling factor (sub) of the input image during testing, and we include the reported run time of the method in ms (rt).

	Classes		Categories	
	IoU	iIoU	IoU	iIoU
ResNet-38	<b>80.6</b>	57.8	<b>91.0</b>	<b>79.1</b>
PSPNet	80.2	<b>58.1</b>	90.6	78.2
TuSimple	80.1	56.9	90.7	77.8
RefineNet	73.6	47.2	87.9	70.6
LRR-4x	71.8	47.9	88.4	73.9
FRRN	71.8	45.5	88.9	75.1
Adelaide Context	71.6	51.7	87.3	74.1
Deep Layer Cascade (LC)	71.1	47.0	88.1	74.1
DeepLab v2 CRF	70.4	42.6	86.4	67.7
Dilation 10	67.1	42.0	86.5	71.1
Scale invariant CNN	66.3	44.9	85.0	71.2
SQ	59.8	32.3	84.3	66.0
ENet	58.3	34.4	80.4	64.0
Ours	72.6	45.5	87.9	71.6

Table 4.13: Quantitative results of pixel-level semantic labeling benchmark using the metrics presented in Section 4.3.1. All numbers are given in percent; refer to Table 4.12 for details on the listed methods.



	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
ResNet-38	<b>95.5</b>	<b>86.8</b>	71.1	<b>96.1</b>	75.2	87.6	81.9	69.8	76.7
PSPNet	95.1	86.3	<b>71.4</b>	96.0	73.5	90.4	80.3	<b>69.9</b>	<b>76.9</b>
TuSimple	95.4	85.9	70.5	95.9	<b>76.1</b>	<b>90.6</b>	<b>83.7</b>	67.4	75.7
RefineNet	94.8	80.9	63.3	94.5	64.6	76.1	64.3	62.2	70.0
LRR-4x	95.0	81.3	60.1	94.3	51.2	67.7	54.6	55.6	69.6
FRRN	94.9	81.6	62.7	94.6	49.1	67.1	55.3	53.5	69.5
Adelaide Context	94.1	81.5	61.1	94.3	61.1	65.1	53.8	61.6	70.6
Deep LC	94.2	81.2	57.9	94.1	50.1	59.6	57.0	58.6	71.1
DeepLab v2 CRF	94.2	79.8	59.8	93.7	56.5	67.5	57.5	57.7	68.8
Dilation 10	93.7	78.9	55.0	93.3	45.5	53.4	47.7	52.2	66.0
Scale inv. CNN	92.2	77.6	55.9	90.1	39.2	51.3	44.4	54.4	66.1
SQ	93.0	73.8	42.6	91.5	18.8	41.2	33.3	34.0	59.9
ENet	90.6	65.5	38.4	90.6	36.9	50.5	48.1	38.8	55.4
Ours	94.7	81.2	61.2	94.6	54.5	76.5	72.2	57.6	68.7

Table 4.14: Detailed results of pixel-level semantic labeling benchmark in terms of the [IoU](#) score on the class level. We list results for the classes in the categories *sky*, *human*, and *vehicle*. The remaining categories are listed in Table [4.15](#). All numbers are given in percent; refer to Table [4.12](#) for details on the listed methods.

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain
ResNet-38	<b>98.7</b>	<b>86.9</b>	<b>93.3</b>	<b>60.4</b>	62.9	<b>67.6</b>	75.0	78.7	<b>93.7</b>	<b>73.7</b>
PSPNet	98.6	<b>86.6</b>	93.2	58.1	<b>63.0</b>	64.5	<b>75.2</b>	<b>79.2</b>	93.4	72.1
TuSimple	98.5	85.9	93.2	57.7	61.1	67.2	73.7	78.0	93.4	72.3
RefineNet	98.2	83.3	91.3	47.8	50.4	56.1	66.9	71.3	92.3	70.3
LRR-4x	97.9	81.5	91.4	50.5	52.7	59.4	66.8	72.7	92.5	70.1
FRRN	98.2	83.3	91.6	45.8	51.1	62.2	69.4	72.4	92.6	70.0
Adelaide Context	98.0	82.6	90.6	44.0	50.7	51.1	65.0	71.7	92.0	72.0
Deep LC	98.1	82.8	91.2	47.1	52.8	57.3	63.9	70.7	92.5	70.5
DeepLab v2 CRF	97.9	81.3	90.3	48.8	47.4	49.6	57.9	67.3	91.9	69.4
Dilation 10	97.6	79.2	89.9	37.3	47.6	53.2	58.6	65.2	91.8	69.4
Scale inv. CNN	96.3	76.8	88.8	40.0	45.4	50.1	63.3	69.6	90.6	67.1
SQ	96.9	75.4	87.9	31.6	35.7	50.9	52.0	61.7	90.9	65.8
ENet	96.3	74.2	85.0	32.2	33.2	43.5	34.1	44.0	88.6	61.4
Ours	98.0	81.4	91.1	44.6	50.7	57.3	64.1	71.2	92.1	68.5

Table 4.15: Detailed results of pixel-level semantic labeling benchmark in terms of the [IoU](#) score on the class level. We list results for the classes in the categories *flat*, *construction*, *object*, and *nature*. The remaining categories are listed in Table 4.14. All numbers are given in percent; refer to Table 4.12 for details on the listed methods.

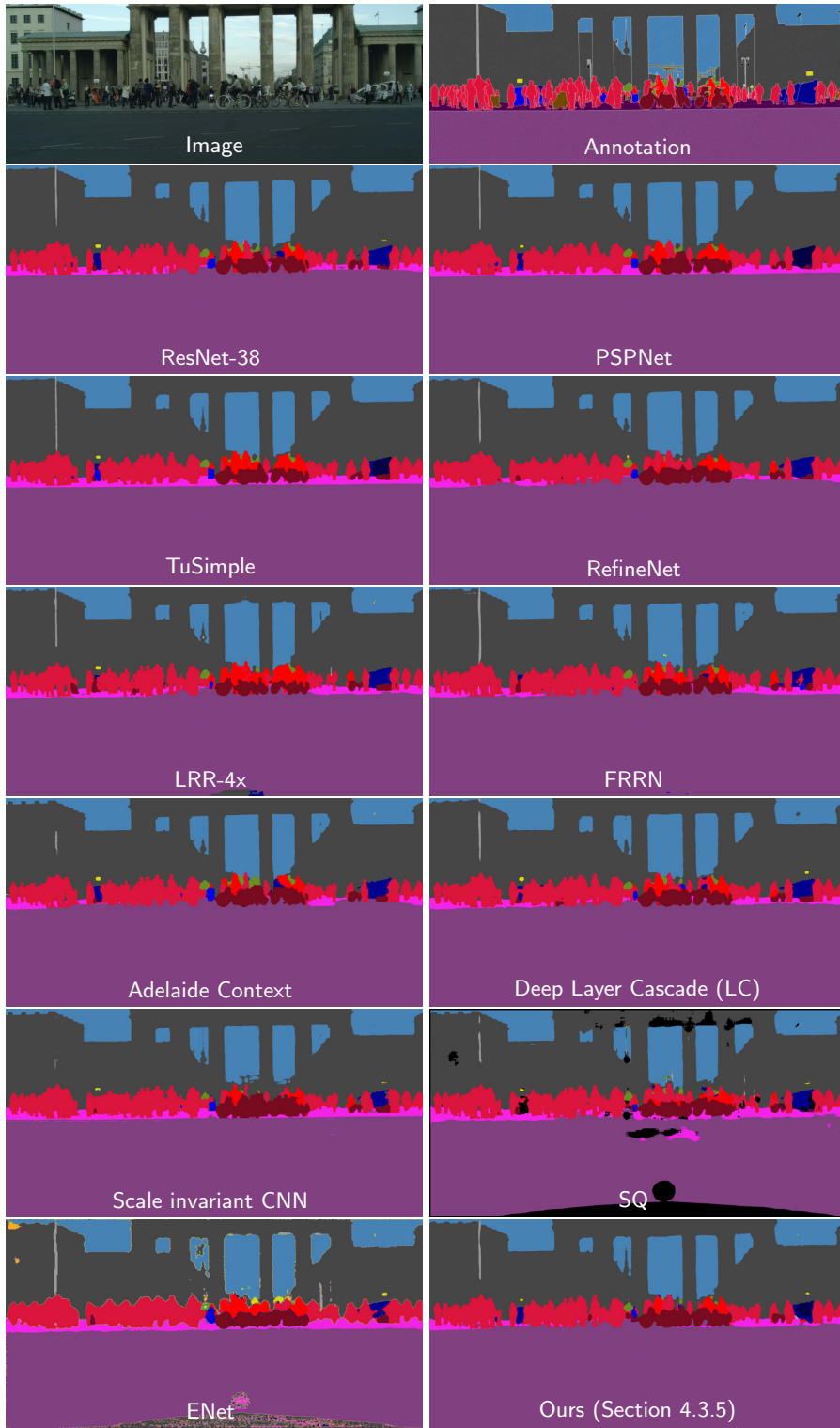


Figure 4.13: Exemplary output of benchmarked methods for the pixel-level semantic labeling task. The image is part of our *test* set and has both, the largest number of instances and persons. Please refer to Table 4.12 for details on the shown methods.

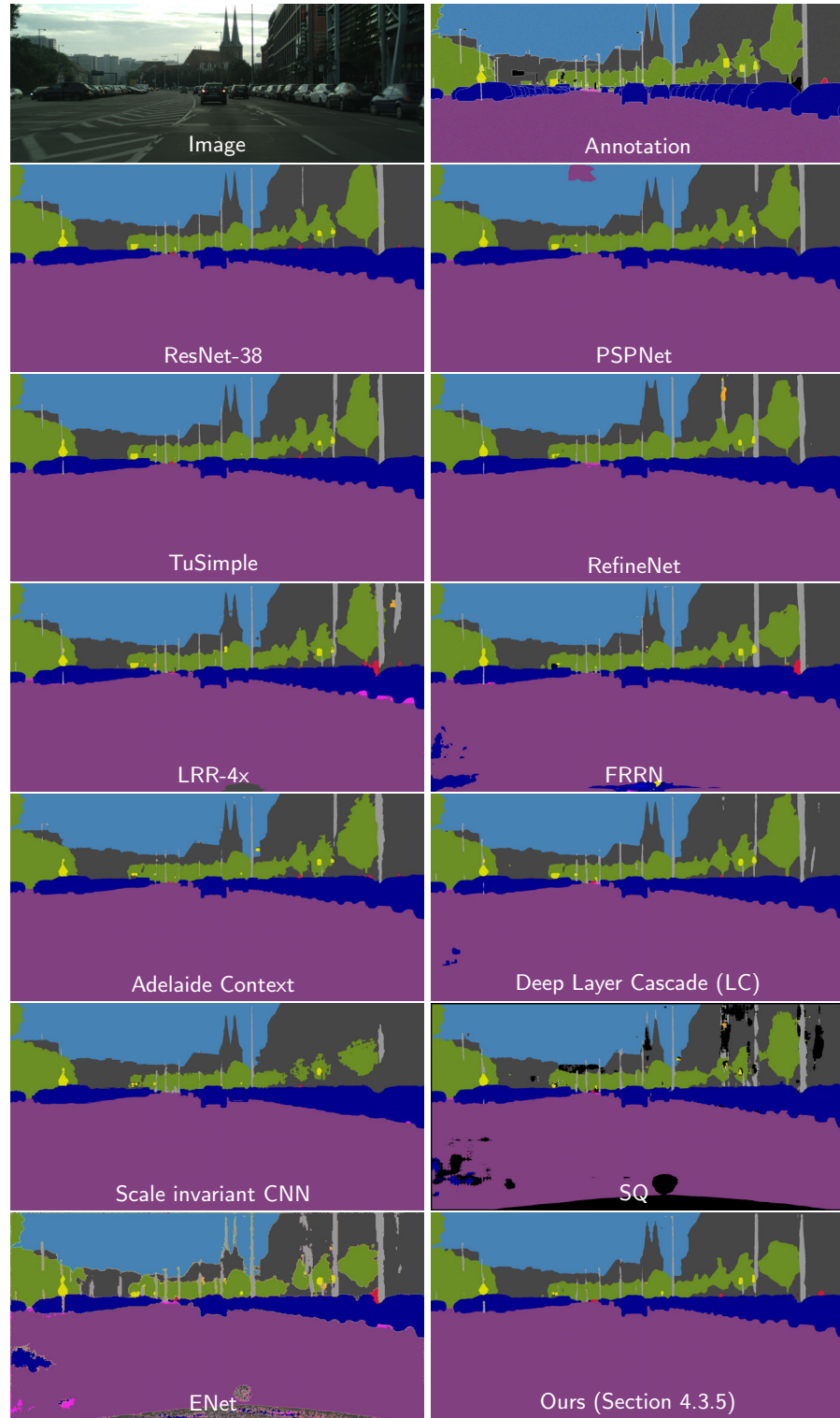


Figure 4.14: Exemplary output of benchmarked methods for the pixel-level semantic labeling task. The image is part of our *test* set and has the largest number of car instances. Please refer to Table 4.12 for details on the shown methods.

However, their performance does not reach the current state of the art as evident from Table 4.13. To the best of our knowledge, so far no researcher designed a network specifically for pixel-level semantic labeling, but nevertheless pretrained on ImageNet to increase generalization capabilities.

Out of all non-anonymous submissions, there is only a single one leveraging the available multi-modal input data, i.e. the *scale invariant CNN* by Kreso et al., 2016, which employs depth cues. However, the depth information is only used to select an appropriate scale out of a set of multi-scale features. There are no deep learning-based features directly extracted on the depth channel and the overall performance is rather low as evident from Table 4.13. All other analyzed methods rely on single LDR images only. A possible explanation is the requirement of an initialization with an ImageNet-like pretrained model paired with the lack of an available equivalent dataset featuring multi-modal data.

Next, we analyze the average performance numbers in Table 4.13 with respect to our different metrics proposed in Section 4.3.1. First, it becomes evident that the class-level and category-level metrics are highly correlated, while the category-level numbers are significantly higher as this is a simpler task. Because the networks are only evaluated in terms of their category-level performance while still predicting on the class-level, we investigated the effects when directly producing category-level results. To this end, we compared the category-level performance of an FCN trained for class-level prediction with the same FCN producing category-level predictions obtained by adding the probability scores prior to computing the argmax. Additionally, we trained an FCN to directly predict category-level labels. Interestingly, we found that all three variants yield performance numbers within a range of 0.3 %. Thus, we conclude that the category-level performance is mainly influenced by the method and hence the network architecture rather than a specific training, which also explains the correlation between class- and category-level metrics. Second, we compare performances in terms of the iIoU metric. We observe that focusing on this score does in fact change the ranking of some methods. Most interesting are *Adelaide Context* (G. Lin et al., 2016) which performs comparably well considering the iIoU and *DeepLab v2 CRF* (L.-C. Chen et al., 2016) which shows a large drop in performance. Both methods leverage a CRF on top of the CNN, which indicates that such a post-processing has large potential in influencing the iIoU metric. In *Adelaide Context*, all CRF potentials are based on an FCN output, while in *DeepLab* the pairwise potentials are simpler regularizers that tend to oversmooth small instances. Similar observations were made based on the initial set of submissions for this benchmark (Cordts et al., 2016), however, there are also exceptions, i.e. the *Scale invariant CNN* by Kreso et al., 2016.

We now consider the  $\text{IoU}$  performance of individual classes as listed in Tables 4.14 and 4.15. We find that the evaluated methods perform best for *road*, followed by *sky* and *car*. Because for pixel-level semantic labeling, each pixel is a (correlated) training example, it is natural that these classes are comparably easy, since they are among the most frequent classes of our dataset, c.f. Figure 4.3. However, this distribution alone cannot explain the overall ranking, in particular *wall*, *fence*, and *truck* are the hardest but not the rarest classes. Instead, we expect typical class-specific properties to influence the overall performance, such as a repeating and hence simple texture for *road* and *sky*. In contrast, a standalone *wall* is easily confused with a *building* or a *fence* might get classified as the object behind it, which can be seen through its holes. Further, we observe that the performance of *truck*, *bus*, and *train* varies strongest across methods. Note that all three classes contain large but rare objects, which hence seems to be a key challenge of the dataset.

Next, we analyze the results in order to identify important design choices that lead to well performing neural networks. When comparing the method properties (Table 4.12) with their average performance (Table 4.13), three observations can be made. First, out of the five best performing approaches, four use the additional coarse annotations for training. All authors also report  $\text{IoU}_{\text{class}}$  performance numbers for the same models trained without the additional data: the smallest gain is 1.2 % (Wang et al., 2017), whereas the largest is 3 % (Ghiasi and Fowlkes, 2016). This clearly confirms the value of the coarse annotations and complements our findings in Section 4.3.3. Second, three of the best four approaches apply the network to multiple image scales for testing and average the obtained results. Z. Wu et al., 2016 report a gain of 0.9 %  $\text{IoU}_{\text{class}}$  by applying this trick. However, for a real-time application as is the focus in this dissertation, multi-scale testing is not a preferred choice since it significantly increases run time. Our last observation is that applying a CRF as post-processing or downscaling the input image potentially harms performance. Only three approaches leverage a CRF and only two apply downscaling and none of them is within the top five methods. This observation confirms our findings based on the initial set of submissions when publishing the dataset (Cordts et al., 2016).

Eventually, we aim to understand the correlation across different methods in terms of their predictions on the Cityscapes dataset. For these experiments, we seek to analyze only the best performing methods on the dataset that have approximately the same performance, in order to reduce the effects induced by different levels of accuracy. To this end, we also include those submissions that have a public entry in the result table, but are anonymous. We then select the ten best performing methods as of April 15<sup>th</sup>, 2017, ranging from 76.5 %  $\text{IoU}_{\text{class}}$  to 80.6 %. Based on this selection, we determine for each pixel

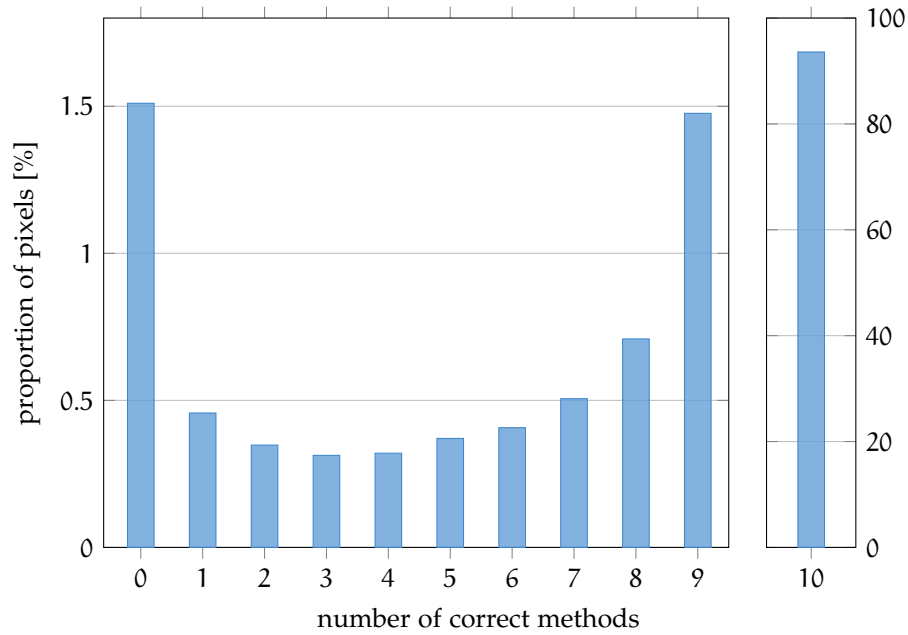


Figure 4.15: Histogram over number of correct methods. We select the ten best performing methods in the pixel-level semantic labeling benchmark and count for each pixel of the test set that is not ignored during evaluation, how many methods correctly predicted the semantic class of that pixel. Based on these counts we then compute the histogram over all pixels and normalize. This evaluation indicates that the methods produce correlated results, as typically all or nearly all methods are either correct or in-correct.



that is not ignored during evaluation, how many methods were correct and compute a histogram over these counts over all evaluated pixels, c.f. Figure 4.15. This evaluation shows that 93.6 % of the pixels are correctly classified by all methods. Out of the remaining pixels, 23 % are misclassified by a single method only and another 23 % are misclassified by all methods; we denote those pixels as *hard pixels* in the following. These observations indicate that different networks trained and evaluated on the same dataset produce correlated results. Since the analysis so far is based on global statistics, we now address the importance of the hard pixels for the overall  $\text{IoU}_{\text{class}}$ , where the contribution of individual classes is balanced. To this end, we re-evaluate the best method (*ResNet-38* by Z. Wu et al., 2016) while ignoring the hard pixels. In doing so, the performance increases from 81.4 %  $\text{IoU}_{\text{class}}$  to 88.1 %. When instead ignoring the erroneous but not hard pixels, performance reaches the same level, i.e. 89.5 %, indicating that the hard pixels account for approximately half of the overall performance loss. These findings motivate the question, if we can combine the predictions of a set of different methods to obtain better results, similar to ensemble models that became popular for ILSVRC (Russakovsky et al., 2015). The concept of such models is to train multiple neural networks for the same task with different initializations and to combine their results at test time. In our case, we build our ensemble out of the  $N$  best submissions and use their majority vote for each pixel as prediction. Such an ensemble has the advantage of consisting of a highly diverse set of models; however, these models have a varying individual performance. To compensate for the latter, we tried all values between 3 and 10 for the number of models  $N$ . We found that in all cases, the performance increases with respect to the best single model and combining the top five methods yields the maximum, i.e. 81.4 %. Since this performance is only a small improvement compared with the best single model (80.6 %), we conclude that even though ensemble models offer a potential for improvements in theory, it is challenging to actually realize a significant margin in practice.

#### 4.3.5 Real-time fully convolutional networks

The scope of this dissertation is to work towards an efficient system for urban semantic scene understanding, c.f. Section 1.4. To this end, we aim for excellent recognition performance at real-time speeds. However, as evident from Section 4.3.4, the methods that report a run time are either rather slow or have significant draw-backs in terms of accuracy. For those approaches that do not report run time values, we can assume a rather high run time by analyzing the CNN architecture and method details. In this section, we aim to overcome these shortcomings and combine existing approaches from the literature in



order to obtain an efficient yet high-performing model for pixel-level semantic labeling.

Most state-of-the-art approaches for pixel-level semantic labeling are based on a Fully Convolutional Network (FCN), c.f. Sections 2.1.2 and 4.3.4, which means that the methods are centered around a CNN, where all layers have a finite receptive field. This CNN is typically pretrained on ImageNet for the image classification task, e.g. VGG (Simonyan and Zisserman, 2015) in the work of Shelhamer et al., 2017 or ResNet (K. He et al., 2016) in the more recent approach by Zhao et al., 2017. In this section, we follow the described approach and center our experiments around a single FCN that is pretrained on ImageNet. In order to select the network architecture, we conduct a preliminary experiment, where we measure the run time of different models and list the performances of the models on the ILSVRC'12 validation set as reported in the individual publication, c.f. Table 4.16. For timing, we convert the models to an FCN architecture where necessary (c.f. Shelhamer et al., 2017) and run the network on 1.2 MP RGB images. The selected input size is small enough such that all inspected models fit within the available GPU memory and is large enough such that they can be obtained by cropping the lower and upper image parts without losing relevant information, c.f. Figure 4.1. Besides these modifications, we do not alter or train the networks; especially we do not include any upscaling or post-processing in our timings. As evident from Table 4.16, GoogLeNet v1 (Szegedy et al., 2015) yields an excellent tradeoff between run time and performance and is therefore selected as the base network for the following experiments.

The GoogLeNet architecture proposed in Szegedy et al., 2015 is intended for image classification, where the output is a score vector of as many elements as classes. For pixel-level semantic labeling, we seek for an output that has the same spatial resolution as the input image and assigns such a score vector to each pixel. Further, the label set that we are interested in differs from the one in ImageNet. Therefore, we remove the final inner product and average pooling layers to prepare the network for the new task. As with many other network architectures (Shelhamer et al., 2017), the GoogLeNet has an output stride of 32, meaning that due to strides in convolution or pooling layers the spatial resolution of its output is downsampled by a factor of 32 in both image axes. Since this downscaling is too severe for an accurate segmentation, researchers proposed various approaches to obtain the desired resolution; refer to Chapter 2 for details. In this work, we follow Shelhamer et al., 2017 and opt for skip layers as these are computationally very efficient, simple to use, and integrate well into many network architectures.

For training our network, we start with the same training parameters as in Section 4.3.3, except that we directly train the 8s variant with a single cross-entropy loss, which we found to cause no harm

Network	Ref.	Top-5 Error [%]		Run time [ms]
		Single	Multi-Crop	
SqueezeNet v1.1	[1]	19.7	-	13
NiN	[2]	-	18.2 <sup>b</sup>	22
AlexNet	[3]	19.6 <sup>a</sup>	18.2	41
GoogLeNet v1	[4]	10.1	7.9	48
ResNet-50	[5]	-	5.3	133
GoogLeNet v3	[6]	5.6	4.2	136
ResNet-101	[5]	-	4.6	221
VGG-19	[7]	8.0	7.1	271
VGG-16	[7]	8.1	7.2	296
ResNet-152	[5]	-	4.5	320

[1] Iandola et al., 2016

[2] M. Lin et al., 2014

[3] Krizhevsky et al., 2012

[4] Szegedy et al., 2015

[5] K. He et al., 2016

[6] Szegedy et al., 2016b

[7] Simonyan and Zisserman, 2015

<sup>a</sup> [github.com/BVLC/caffe/tree/master/models/bvlc\\_reference\\_caffenet](https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet)

<sup>b</sup> Estimated by comparing ImageNet performance (top-1 multi-crop 40.64 %, [gist.github.com/mavenlin/d802a5849de39225bcc6#file-readme-md](https://gist.github.com/mavenlin/d802a5849de39225bcc6#file-readme-md)) with AlexNet (40.7 %)

Table 4.16: Comparison of networks in terms of ImageNet classification performance and run time. We list the top-5 error rate without external training data on the ILSVRC’12 validation set as provided in the individual publications if not state otherwise. If available, we report the performance of a single model evaluated on a single image crop, which is closest to our setup. In addition, we provide the performance of a single model tested on multiple crops. Run time is determined by converting the individual models to FCN models (where necessary) and evaluating on RGB images with 1.2 MP. We use Caffe (Jia et al., 2014) with NVIDIA cuDNN 5.1 to run the networks on an NVIDIA Titan X (Maxwell) GPU.

	train	coarse	lr	it	Classes		Categories	
					IoU	iIoU	IoU	iIoU
Base GoogLeNet	✓		1.2	910	64.2	43.2	85.5	69.2
+ Higher learning rate	✓		10	140	65.3	43.6	85.6	68.5
+ Data augmentation	✓		10	812	68.5	48.3	86.9	72.9
+ Include coarse labels	✓	✓	33	1642	72.5	50.6	87.6	71.5
+ Context modules	✓	✓	33	1621	73.9	49.7	87.6	70.7

Table 4.17: Quantitative results of different FCN variants based on the GoogLeNet network architecture (Szegedy et al., 2015). Performance is evaluated on the Cityscapes validation set using the metrics presented in Section 4.3.1 and given in percent. We also indicate the used training data for each method, i.e. *train fine*, and *coarse extra* and list the learning rate (lr) as multiples of  $1 \times 10^{-11}$  as well as the number of training iterations (it) as multiples of  $1 \times 10^3$ . Please refer to the text for details about the individual variants.

when based on GoogLeNet. Further, we train only on the training set and monitor the performance on the validation set. After convergence, the network reaches a performance of 64.2 % IoU, c.f. Table 4.17. This performance is slightly worse than the VGG-based FCN-8s in Section 4.3.3 (65.3 % IoU), c.f. Table 4.9.

Starting from this initial experiment, we now aim to increase the classification performance by optimizing the training procedure without affecting the run time, c.f. Table 4.17. To this end, we repeat the training with increasing learning rates until the training diverges. The maximum converging learning rate yields an improvement of 1.7 % IoU while the training is sped up by a factor of 6.5. Next, we apply data augmentation as common for training deep neural networks, e.g. K. He et al., 2016; Krizhevsky et al., 2012; Zhao et al., 2017. We apply random left-right flipping, RGB color shifts, additive Gaussian noise, Gaussian blur, and affine warping. In doing so, the training takes longer, but the classification performance increases by another 4.9 %. Next, we intend to leverage the coarse annotations that were successfully used by several approaches, c.f. Section 4.3.4. These annotations provide labels for many pixels but cannot aid the precise segmentation, c.f. Figures 4.11 and 4.12. In order to ensure that the gradient during training is balanced with respect to classification (coarse and fine) and segmentation (fine), we use a batch size of 2 consisting of a fine and coarse data pair each. Furthermore, we increase the learning rate by a factor of 3 without losing convergence. Due to the increased amount of training data paired with the extensive data augmentation, the number of iterations needed for convergence doubles and due to a doubled batch size the training time quadru-

ples. Overall, by leveraging the coarse labels we obtain another 5.8 % performance gain.

While the receptive field of GoogLeNet is as large as  $907 \times 907$  pixels (Shelhamer et al., 2017), the effective field is considerably smaller (Luo et al., 2016). Thus, in our last model, we aim to increase the receptive field of the network and in turn the context information that the network can exploit when classifying a pixel. A common approach towards this goal is to use feature representations at multiple scales, e.g. Farabet et al., 2012; Yu and Koltun, 2016; Zhao et al., 2017. We follow this line of research and opt for the context modules proposed by Yu and Koltun, 2016 as parameterized for the Cityscapes dataset. In that work, a set of dilated convolution is appended to a VGG-16 network such that context information at various scales is aggregated. In addition, Yu and Koltun, 2016 modify the FCN and use dilated convolutions instead of strided pooling layers such that the network’s output stride is 8 instead of 32. Since the latter modifications significantly increase the run time, we continue to use the more efficient skip connections for upscaling. Adding the context module to the GoogLeNet FCN yields a performance gain of 1.9 % at an increase of run time of only 6.3 %. However, note that this comes at the cost of a slight decrease of performance with respect to the *iIoU*, which can be attributed to the context modules serving as regularizer that suppresses the detection of small objects. Overall, we achieve a performance of 73.9 % *IoU* on the validation and of 72.6 % on the test set, c.f. Table 4.13. This performance is below of the state of the art, but is the best-performing real-time network by a significant margin.

#### 4.3.6 Cross-dataset evaluation

In order to show the compatibility and complementarity of Cityscapes regarding related datasets, we applied an FCN model trained on our data to CamVid (Brostow et al., 2009) and two subsets of KITTI (Ros et al., 2015; Sengupta et al., 2013). We use the half-resolution model (c.f. Section 4.3.3) to better match the target datasets, but we do not apply any specific training or fine-tuning. In all cases, we follow the evaluation of the respective dataset to be able to compare to previously reported results (Badrinarayanan et al., 2017; Vineet et al., 2015). The obtained results in Table 4.18 show that our large-scale dataset enables us to train models that are on a par with or even outperforming methods that are specifically trained on another benchmark and specialized for its test data. Further, our analysis shows that our new dataset integrates well with existing ones and allows for cross-dataset research.

Dataset	Best reported result	Our result
CamVid <sup>a</sup>	71.2 <sup>b</sup>	72.6
KITTI <sup>c</sup>	61.6 <sup>b</sup>	70.9
KITTI <sup>d</sup>	82.2 <sup>e</sup>	81.2

<sup>a</sup>Brostow et al., 2009

<sup>b</sup>Badrinarayanan et al., 2017

<sup>c</sup>Ros et al., 2015

<sup>d</sup>Sengupta et al., 2013

<sup>e</sup>Vineet et al., 2015

Table 4.18: Quantitative results (average recall in percent) of our half-resolution FCN-8s model trained on Cityscapes images and tested on CamVid and KITTI.

#### 4.4 INSTANCE-LEVEL SEMANTIC LABELING<sup>4</sup>

In Section 1.2, we introduced common tasks for semantic scene understanding. Pixel-level semantic labeling, which we discussed in Section 4.3 and which is the focus of this dissertation, does not aim to separate individual object instances. The instance-level challenge of the Cityscapes benchmark goes beyond and requires algorithms to produce segments of individual traffic participants in the scene. Training and evaluation of such methods is enabled by the instance-level annotations of humans and vehicles, c.f. Section 4.2.2. As this task goes beyond the scope of this dissertation, we briefly present the task and its metrics in Section 4.4.1, but omit a discussion of the state of the art.

##### 4.4.1 Metrics

For the task of instance-level semantic labeling, approaches need to produce predictions of traffic participants in the image. Each such detection consists of a pixel-level segmentation mask, a semantic class label, and a confidence score. Note that these detections can potentially overlap, which needs to be taken into account by any evaluation metric.

To evaluate the performance of a method, we center our metric around the region-level average precision ( $AP^r$ ; Hariharan et al., 2014) that we compute for each class while averaging across a range of different overlap thresholds. Specifically, we start by matching predictions with ground truth instances of the considered semantic class. A pair of such regions matches, if they overlap by at least a given thresh-

<sup>4</sup> This section is based on the contributions of Mohamed Omran with respect to the instance-level semantic labeling benchmark (Section 4 in Cordts et al., 2016). It is included in this dissertation for completeness.

old, while the overlap is defined as the Intersection-over-Union (IoU), c.f. Figure 3.5. If this overlap threshold is at least 50 %, it holds that no instance prediction can have multiple matches with the ground truth, since the latter is non-overlapping. Based on these matches, we define a predicted instance that matches the ground truth as a True Positive (TP) and as a False Positive (FP) otherwise. Multiple matches with the same ground truth instance are also counted as False Positives (FPs). Unmatched ground truth instances are False Negatives (FNs).

After counting occurrences of the different cases, the precision is defined as

$$\text{Precision} = \frac{\sum \text{TP}}{\sum \text{TP} + \sum \text{FP}} \quad (4.3)$$

and denotes the proportion of the predicted instances that are correct. Further, the recall is defined as

$$\text{Recall} = \frac{\sum \text{TP}}{\sum \text{TP} + \sum \text{FN}} \cdot \quad (4.4)$$

and denotes the proportion of ground truth instances that are detected. Since a sensible evaluation should not be restricted to a single operating point, a varying threshold on the confidence score of the instance predictions is applied. If this threshold has a rather low value, the recall becomes high, but the precision decreases and vice versa for high values. If we choose all occurring confidence values as thresholds, we can compute the precision-recall curve, where the precision is plotted against the recall with the confidence threshold as an implicit parameter. The area under this curve is the Average Precision (AP) and the higher this value, the better the method.

In terms of the value of the overlap threshold, we follow T.-Y. Lin et al., 2014 and use 10 different overlaps ranging from 0.5 to 0.95 in steps of 0.05. We repeat the above analysis for each of these values, average the resulting APs, and denote the result again as AP for simplicity. In doing so, we can prevent to give advantage to methods that perform particularly well at a single overlap value.

Eventually, the main evaluation metric is obtained by averaging over the evaluated classes, yielding the mean average precision AP. In addition, we compute three minor scores. First, we restrict the evaluation to an overlap value of 50 % yielding  $\text{AP}^{50\%}$ . Second and third, we only consider traffic participants that are within 100 m and 50 m distance from the camera, denoted as  $\text{AP}^{100\text{m}}$  and  $\text{AP}^{50\text{m}}$ , respectively. These minor scores allow for an evaluation that focuses on different domains important for autonomous driving. Often an accurate segmentation is not required as reflected by  $\text{AP}^{50\%}$  and objects may have a different importance depending on their distance to the vehicle as assessed by  $\text{AP}^{100\text{m}}$  and  $\text{AP}^{50\text{m}}$ .

## 4.5 BENCHMARK SUITE

In Sections 4.3 and 4.4, we described the two major tasks within the Cityscapes benchmark suite. Regarding both tasks, the purpose of the benchmark suite is to be able to easily, transparently, and fairly compare and rank different methods. To this end, we followed the three concepts that are described in the following paragraphs and are consistent with previous major datasets, e.g. ImageNet/[ILSVRC](#) (Russakovsky et al., 2015), [PASCAL VOC](#) (Everingham et al., 2014), Microsoft [COCO](#) (T.-Y. Lin et al., 2014), or [KITTI](#) (Geiger et al., 2013).

**PUBLIC EVALUATION SCRIPTS** For evaluating results of the different tasks in our benchmark, we implemented corresponding evaluation scripts. We published<sup>5</sup> these scripts together with the dataset such that the exact evaluation protocol is transparent, the scripts can be reviewed by other researchers, and authors can use exactly the same scripts to evaluate their methods on the validation set. As additional benefits, the scripts could be used for future projects, such that ideally the comparability between performances on different datasets is given.

**EVALUATION SERVER** Based on the evaluation scripts, we implemented an evaluation server<sup>6</sup>, where users can upload their results such that these are automatically evaluated. In doing so, we can offer a convenient service that ensures the same evaluation scheme for all users and produces performance scores that can be fairly compared. In addition, the evaluation server allows us to keep the test set annotations private. Users can only access the input data for their methods, e.g. the images, run their method on this data and upload the predictions for evaluation, but they cannot conduct evaluations on the test set on their own. In doing so, we can control how often a method can be evaluated on the test set and therefore effectively prevent an over-fitting, which in turn enables a realistic comparison.

Ideally, a single method would be restricted to a single evaluation run only, since this best reflects the real-world requirements for a system for urban scene understanding. An intelligent vehicle equipped with such a system needs to parse incoming images and decide on their content without getting a second chance. However, in practice researchers develop different variants of a method that they would like to compare, especially prior to conference submission deadlines; the performance of a method might increase over time; or there might be bugs in early implementations. Thus, a sensible amount of submissions of the same method should be allowed. Further, the evaluation server can only restrict submissions based on user accounts and not

<sup>5</sup> [www.github.com/mcordts/cityscapesScripts](http://www.github.com/mcordts/cityscapesScripts)

<sup>6</sup> [www.cityscapes-dataset.com/submit](http://www.cityscapes-dataset.com/submit)



based on method types, since it has no knowledge about the latter. Therefore, we restrict submissions to one every 48 h per user and a maximum of 6 submissions per month. Additionally, we manually control that multiple users with the same or close affiliation do not circumvent this restriction. Overall, automated parameter tuning and overfitting is effectively prevented by our evaluation server and simultaneously the requirements mentioned above are satisfied.

**RANKING WEBSITE** Once submissions are evaluated by the evaluation server, users have the option to publish their results in the official benchmark table<sup>7</sup>. This table provides an overview of various results on the Cityscapes dataset and allows to fairly compare different methods. Besides the performance metrics, the table also contains meta data that the participants provided for their submission, such as the input data used by the methods, the leveraged training data, or the method’s run time. To satisfy our goal of transparency, we also provide the history for each method, in case the performance numbers were updated over time.

#### 4.6 IMPACT AND DISCUSSION

In this chapter, we introduced and analyzed Cityscapes, a large-scale dataset paired with a benchmark suite focused on semantic urban scene understanding. We created the largest dataset of street scenes with high-quality and coarse annotations to date. While doing so, we paid particular attention to the needs of autonomous driving applications. Motivated by our findings in Chapter 3, we ensured that typical multi-modal input data is available. Further, we recorded in 50 different cities to assess generalization capabilities. The proposed dataset is extensively studied to provide statistics that are compared with related datasets and confirm the unique characteristics of Cityscapes. Based on our dataset, we developed benchmarks and evaluation methodologies that reflect both established metrics in the research community as well as novel ones tailored to autonomous driving. The benchmark allows us to fairly compare state-of-the-art approaches for semantic scene understanding. Based on our findings, we developed an efficient FCN for pixel-level semantic labeling.

Since the publication of Cityscapes, numerous researchers have registered to download the dataset and submitted their results to our evaluation server as shown in Figure 4.16. We observe a more than linearly increasing number of registered users over time and note that many researchers downloaded the data shortly after release. We account this to the fact that we announced the dataset prior to its public release (Cordts et al., 2015) and created a mailing list to inform interested people of its state. The number of submissions increases

<sup>7</sup> [www.cityscapes-dataset.com/benchmarks](http://www.cityscapes-dataset.com/benchmarks)



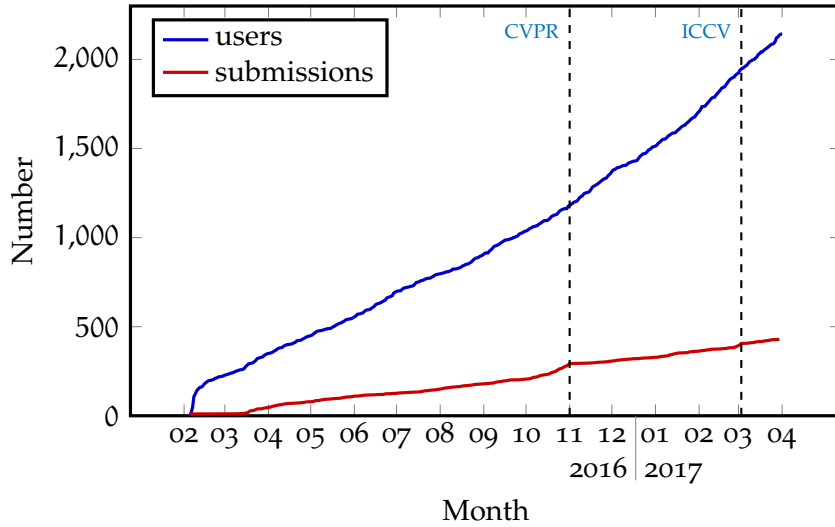


Figure 4.16: Activity statistics for the Cityscapes dataset over time. We show the number of registered users as well as the number of submissions to our evaluation server. In addition, we mark the submission deadlines for the major computer vision conferences [CVPR](#) and [ICCV](#). Note that the dataset was realeased on February 20<sup>th</sup>, 2016 with an initial set of 10 submissions. Ever since we observe a more than linearly increasing number of registered users. The number of submissions increases approximately linearly, with additional submissions prior to the submission deadlines of [CVPR](#) and [ICCV](#). These statistics include registered users and submissions prior to April 15<sup>th</sup>, 2017.

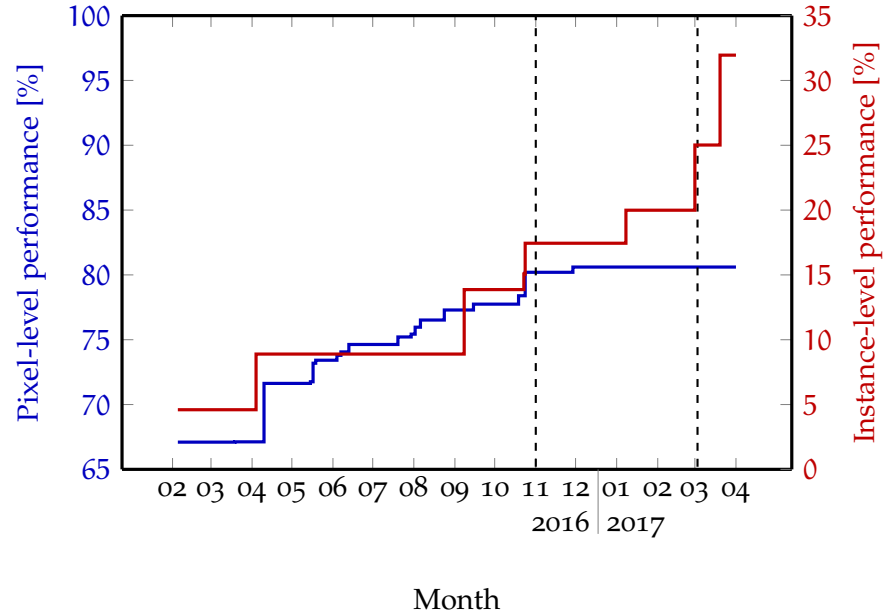


Figure 4.17: Best performing methods on the Cityscapes dataset over time. We show the best  $\text{IoU}$  value for the pixel-level challenge and the best  $\text{AP}$  value for the instance-level task at the respective points in time. The ranges of the y-axes are chosen to cover the same absolute range of performance. Note that we only consider publicly visible submissions, but use the dates of the submissions in order to generate this plot, not the publication dates. We observe that considerable progress has been made in both tasks, while more recently the improvements in the instance-level task are more significant. These statistics include all submissions prior to April 15<sup>th</sup>, 2017.

approximately linearly with an additional increase shortly before submission deadlines to major computer vision conferences. As of April 15<sup>th</sup>, 2017, i.e. one year and two months after release, there are 2144 registered researchers and 429 submissions to our evaluation server. These statistics correspond to 5.1 registrations and 1 submission on average per day.

Enabled by the large-scale Cityscapes benchmark, our research community achieved considerable advances in semantic urban scene understanding. Figure 4.17 documents this progress and shows the performance of the best method on the Cityscapes leaderboard over time. Prior to the publication, we already received submissions from independent research groups running their state-of-the-art methods for pixel-level semantic labeling at that time on our dataset. The best method (Yu and Koltun, 2016) achieved a performance of 67.1 % **IoU**. Ever since, performance constantly increased, reaching 80.6 % with the method of Z. Wu et al., 2016. Considering the instance-level task, progress is even larger. Starting with the baseline performance of 4.6 % (Cordts et al., 2016), the best method (K. He et al., 2017) has increased this number by a factor of 7, achieving 32 % **AP**. Compared to the pixel-level labeling challenge, we account the larger advances in the instance-level task to the fact that this topic is newer, less extensively studied, and gains an increasing popularity. In addition, the baseline performance at the release of the benchmark was considerably lower, simplifying the achievement of larger relative margins.



## CONTENTS

5.1	Related Scene Representations . . . . .	110
5.1.1	Unsupervised image segmentation . . .	110
5.1.2	Street scene environment models . . . .	111
5.1.3	Layered environment models . . . . .	111
5.2	The Stixel Model . . . . .	112
5.3	Graphical Model . . . . .	113
5.3.1	Prior . . . . .	115
5.3.2	Data likelihood . . . . .	117
5.4	Inference . . . . .	120
5.4.1	Dynamic programming . . . . .	121
5.4.2	Algorithmic simplification . . . . .	122
5.4.3	Approximations . . . . .	123
5.5	Parameter Learning . . . . .	124
5.6	Object-level Knowledge . . . . .	125
5.6.1	Extended Stixel model . . . . .	126
5.7	Experiments . . . . .	129
5.7.1	Datasets . . . . .	131
5.7.2	Metrics . . . . .	131
5.7.3	Baseline . . . . .	132
5.7.4	Impact of input cues . . . . .	132
5.7.5	Impact of approximations . . . . .	135
5.8	Discussion . . . . .	136

In recent years, more and more vision technology has been deployed in vehicles, mainly due to the low cost and versatility of image sensors. As a result, the number of cameras, their spatial resolution, and the list of algorithms that involve image analysis are constantly increasing. This poses significant challenges in terms of processing power, energy consumption, packaging space, and bandwidth, all of which are traditionally limited on embedded hardware commonly found in a vehicle.

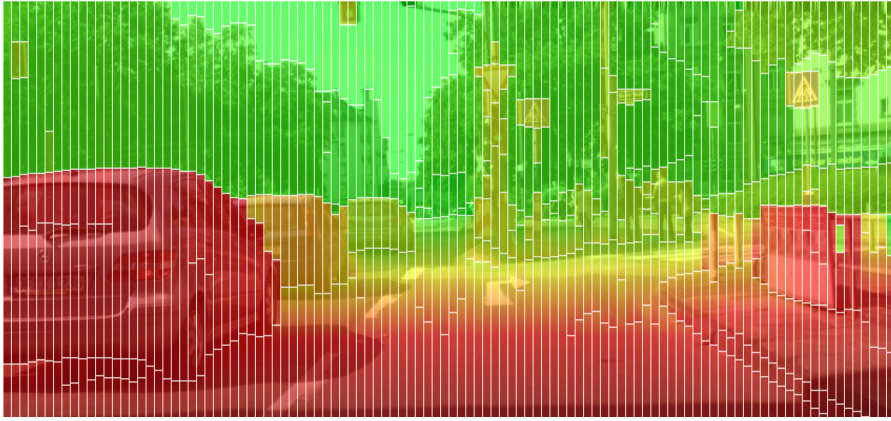
In light of these constraints, it becomes clear that an essential part of modern vision architectures in intelligent vehicles is concerned with sharing of resources. One way towards this goal is to compress the scene into a compact representation of the image content that abstracts the raw sensory data, while being neither too specific nor too generic, so that it can simultaneously facilitate various tasks such as object detection, tracking, segmentation, localization, and mapping.

This representation should allow for structured access to depth, semantics, and color information and should carry high information content while having a low memory footprint to save bandwidth and computational resources.

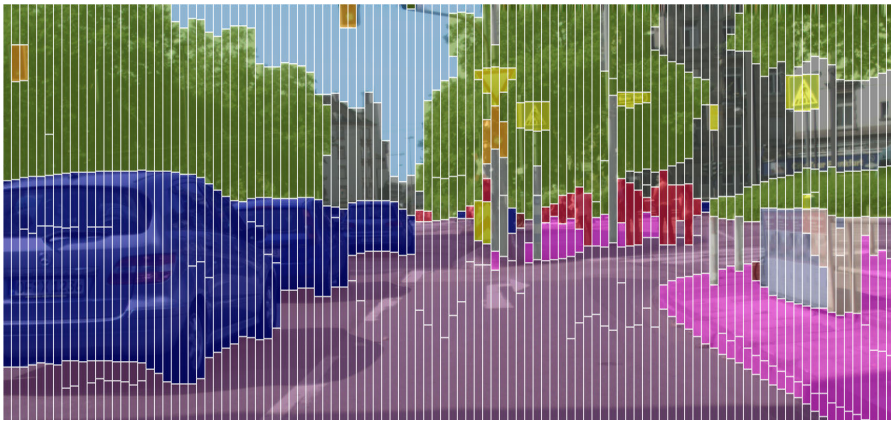
Such a scene representation is obtained by the Stixel World as first proposed by Badino et al., 2009 and extended by Pfeiffer and Franke, 2011b. Stixels are specifically designed for the characteristic layout of street scenes and are based on the observation that the geometry is dominated by planar surfaces. Moreover, prior knowledge constrains the structure of typical street scenes predominantly in the vertical image direction, i.e. objects are on top of a supporting ground plane and the sky is typically in the upper part of the image. Furthermore, there is a characteristic depth ordering from close to far away along this direction. Note that those geometric relations are less pronounced in the horizontal domain. Therefore, the environment is modeled as a set of Stixels: thin stick-like elements that constitute a column-wise segmentation of the image, see Figure 5.1a. Stixels are inferred by solving an energy minimization problem, where the unary terms are driven by depth information via stereo vision and structural and semantic prior information is taken into account to regularize the solution. Overall, they provide a segmentation that contains all relevant information at a sufficient level of detail while being compact and easily parsable for downstream automotive vision applications, e.g. Benenson et al., 2012; Enzweiler et al., 2012; Erbs et al., 2012; Franke et al., 2013; Xiaofei Li et al., 2016; Muffert et al., 2014; Pfeiffer and Franke, 2011a. The main benefits for such subsequent components are that using Stixels as primitive elements either decreases parsing time, increases accuracy, or even both at the same time. Also in Chapter 3, we saw that a scene parsing system based on Stixel superpixels yields best scene recognition performance at real-time speeds.

A second way to share computational resources is outlined by deep neural networks, where dedicated features for specialized tasks share a common set of more generic features in lower levels of the network. In Chapter 4, we learned that a pixel-level labeling based on deep learning methods and paired with a sufficient amount of training data is very powerful and leads to state-of-the-art recognition performance. Especially, when tuned for real-time speeds as in Section 4.3.5, deep networks become highly attractive for automotive applications.

Thus, the goal of this chapter is to work towards the scene representation called *Semantic Stixels*, c.f. Figure 5.1, combining the strengths of Stixels for representing urban scenes with pixel-level CNNs for a semantic scene parsing. To this end, we start by discussing related scene representations in Section 5.1. We continue by presenting the underlying world assumptions in Section 5.2 as originally proposed by Pfeiffer and Franke, 2011b. We then formalize the Stixel model via a Conditional Random Field (CRF) in Section 5.3. Based on approx-



(a) Depth representation, where Stixel colors encode disparities from close (red) to far (green).



(b) Semantic representation, where Stixel colors encode semantic classes following Chapter 4.

Figure 5.1: Scene representation obtained via Semantic Stixels. The scene is represented via its geometric layout (top) and semantic classes (bottom).

imations and ideas presented by Pfeiffer, 2011, an efficient inference scheme respecting the formalism of the CRF paired with a complexity analysis is derived in Section 5.4. Again leveraging the CRF, we outline an automated parameter learning in Section 5.5. Subsequently, we show that we can exploit the CRF formulation and extend the Stixel model with additional data and prior information, e.g. driven by object detectors as sketched in Section 5.6. Eventually, the Semantic Stixel World is evaluated (Section 5.7) and we conclude with a discussion in Section 5.8.

This chapter is based on the journal article of Cordts et al., 2017b and contains verbatim quotes. The article was published as joint work with Timo Rehfeld and Lukas Schneider and unifies previous work on the Stixel World. Timo Rehfeld proposed to enrich the Stixel segmentation with semantic information via pixel-level input cues as first presented in (Scharwächter and Franke, 2015). Lukas Schneider

improved this methodology and made the inference tractable for a large number of semantic classes (L. Schneider et al., 2016). The contribution relevant for this dissertation is the formalism of the Stixel World via a graphical model that allows to unify previous variants into a single model, to provide an analysis of the inference scheme, and to learn the model parameters from training data. A first version of the graphical model was presented in Cordts et al., 2014, paired with an extension with object-level cues that we address in Section 5.6.

## 5.1 RELATED SCENE REPRESENTATIONS

Depending on the perspective, the Semantic Stixel World is related to different lines of research. Considering its output, the Stixel segmentation is related to stereo vision and semantic labeling. However, as this output directly depends on the pixel-level input, a discussion of methods for stereo vision goes beyond the scope of this dissertation and the reader is referred to excellent benchmarks such as KITTI (Geiger et al., 2013). Related work for semantic labeling has already been covered in Chapter 2 and a benchmark is provided in Section 4.3.4.

Interpreting the Stixels as a segmentation of the input image based on various cues, we cover unsupervised image segmentation in Section 5.1.1. Simultaneously, the Stixel model is designed as an environment model for autonomous driving in street scenes. Thus, we address related representations with the same scope in Section 5.1.2. In Section 5.1.3, we turn towards methods exploiting similar geometric model assumptions about the environment, i.e. a layered structure of *support*, *vertical*, and *sky* regions.

This chapter extends the Stixel World proposed by David Pfeiffer in Pfeiffer, 2011; Pfeiffer and Franke, 2011b. There, the environment model, data and prior terms, as well as inference via Dynamic Programming (DP) were introduced. We extend this work with a graphical model, providing a clear formalism, we additionally leverage semantic and color information, we analyze the inference scheme, and we outline automatic parameter learning. Note that there exist several other Stixel-like models implemented by different research groups, i.e. Benenson et al., 2012; Levi et al., 2015; M.-Y. Liu et al., 2015; Rieken et al., 2015; Sanberg et al., 2014.

### 5.1.1 Unsupervised image segmentation

The task of unsupervised image segmentation is to segment an image into regions of similar color or texture. Approaches in this field, include superpixel methods, e.g. Achanta et al., 2012; Comaniciu and Meer, 2002; Felzenszwalb and Huttenlocher, 2004; Levinshtein et al.,



2009, or those producing segmentation hierarchies or large regions (Arbeláez et al., 2011; Carreira and Sminchisescu, 2012). Typically, these unsupervised methods rely on simple model constraints and the resulting segment size is controlled by only a few parameters. Hence, these methods are fairly generic and are often leveraged as the smallest element of processing granularity in contrast to directly accessing individual pixels. In doing so, efficiency can be increased and classification is guided, e.g. for object detection (Girshick et al., 2014; Uijlings et al., 2013), or for semantic labeling (Arbeláez et al., 2012; Carreira et al., 2012b).

Also Stixels can be interpreted as superpixels providing a segmentation of the input image data. However, in contrast to generic unsupervised segmentation methods, Stixels are specific to street scenes. They exploit prior information and the typical geometric layout to yield an accurate segmentation. Further, Stixels are restricted to rectangular shape and are computed independently for each column. In Chapter 3, we investigated different image segmentation techniques when being used as first processing step in a semantic urban scene understanding system and found that Stixels lead to the best results.

#### 5.1.2 *Street scene environment models*

When representing the environment of an autonomous vehicle, occupancy grid maps are a common choice, e.g. Dhiman et al., 2014; Muffert et al., 2014; Nuss et al., 2015; Thrun, 2002. While the Stixels are inferred in the image plane, grid map models represent the surroundings in bird’s eye perspective and partition the scene into a grid. Occupancy information is then assigned to each grid cell, such that the drivable space, obstacles and occluded areas can be distinguished.

Similar to grid maps, the Stixel segmentation distinguishes between *support* regions, i.e. drivable area, and *vertical* segments, i.e. obstacles. Furthermore, it allows for an explicit reasoning about non-observable elements of the scene. Overall, transferring the Stixel World into an occupancy grid is straight-forward.

#### 5.1.3 *Layered environment models*

In our Stixel model, we distinguish between three structural classes, i.e. *support*, *vertical* and *sky*, to represent the environment. An approach to infer the geometric surface layout of elements in an image based on exactly these three classes was previously proposed by Hoiem et al., 2007. The method relies on features from different cues, such as color, texture, superpixel location and shape, as well as perspective information. Hoiem et al., 2007 further classify vertical segments into different properties such as porous or solid, while we classify Stixel segments into different semantic classes.

Felzenszwalb and Veksler, 2010 introduce the tiered scene labeling problem, a model where the scene is vertically split into a bottom, middle, and top part, for example reflecting ground, object, and sky regions. The middle part is then further subdivided horizontally into segments of varying extent. Restricting the scene to such a tiered layout, the globally optimal solution can be inferred via Dynamic Programming (DP) yielding improved results with respect to Hoiem et al., 2007. Nevertheless, the layout is too restrictive to accurately represent complex street scenes, c.f. Chapter 4. The Stixel model assumes a similar scene structure and exploits such constraints for efficient inference.

More recently, M.-Y. Liu et al., 2015 combined the tiered scene model with the column-wise Stixel segmentation. The approach infers semantic and depth information jointly based on features from a deep neural network and stereo matching, respectively. In contrast, Stixels rely on precomputed semantic and depth cues paired with confidence metrics. While this is in theory disadvantageous compared to the approach of M.-Y. Liu et al., 2015, doing so allows for a pipelining of the individual components and thus increases the achievable frame rate of the overall system. Furthermore, M.-Y. Liu et al., 2015 extend the tiered model to up to four layers, but the method does not scale well to more layers. In contrast, the Stixel model allows for as many layers per column as there are image pixels, enabling the representation of complex scenes.

## 5.2 THE STIXEL MODEL

The Stixel World  $\mathcal{S}$  as proposed by Pfeiffer and Franke, 2011b is a segmentation of an image into superpixels, where each superpixel is a thin stick-like segment with a class label and a 3D planar depth model. The key difference between Stixels and other segmentation approaches is that the segmentation problem of a  $w \times h$  image is broken down to  $w/w_s$  individual 1D segmentation problems, one for each column of width  $w_s$  in the image. The horizontal extent  $w_s \geq 1$  is fixed to typically only a few pixels and chosen in advance to reduce the computational complexity during inference. The vertical extent of each Stixel is inferred explicitly in our model. To further control the run time of our method, we apply an optional downscaling in the vertical direction by a factor of  $h_s \geq 1$ .

The idea behind this approach is that the dominant structure in road scenes occurs in the vertical domain and can thus be modeled without taking horizontal neighborhoods into account. This simplification allows for efficient inference, as all columns can be segmented in parallel. We regard our model as a *medium-level* representation for three reasons: (1) each Stixel provides an abstract representation of depth, physical extent, and semantics that is more expressive than

individual pixels; (2) the Stixel segmentation is based upon a street scene model, compared to bottom-up super-pixel methods; (3) Stixels are not a high-level representation, as individual object instances are covered by multiple Stixels in their horizontal extent. All in all, Stixels deliver a compressed scene representation that subsequent higher-level processing stages can build on.

The label set of the Stixel class labels is comprised of two hierarchical layers. The first contains three *structural classes*, “support” ( $\mathcal{S}$ ), “vertical” ( $\mathcal{V}$ ), and “sky” ( $\mathcal{Y}$ ). All three structural classes can be distinguished exclusively by their geometry and reflect our underlying 3D scene model: support Stixels are parallel to the ground plane at a constant height and vertical/sky Stixels are perpendicular to the ground plane at a constant/infinite distance. In the second layer, we further refine the structural classes to *semantic classes* that are sub-classes of the sets  $\mathcal{S}$ ,  $\mathcal{V}$ , and  $\mathcal{Y}$ . Semantic classes such as road or sidewalk, for example, are in the support set  $\mathcal{S}$ , whereas building, tree or vehicle are vertical, i.e. in  $\mathcal{V}$ . The actual set of semantic classes is highly dependent on the application and is further discussed in the experiments in Section 5.7.

The input data for our inference is comprised of a dense depth map, a color image, and pixel-level label scores for each semantic class. Note that all three input channels are optional and are not restricted to specific stereo or pixel classification algorithms. In Section 5.6, we discuss an extension, where we leverage object detections as additional input channel.

In the following sections, we focus on a single image column of width  $w_s$  and provide a detailed mathematical description of the Stixel model. Our graphical model defines the energy function and gives an intuition on factorization properties and thus statistical independence assumptions. Further, we describe an efficient energy minimization procedure via Dynamic Programming (DP) yielding the segmentation of one image column. Subsequently, we sketch learning of the model parameters from ground truth data via Structured Support Vector Machines (S-SVMs).

### 5.3 GRAPHICAL MODEL

In the following, we define the posterior distribution  $P(\mathbf{S}_\cdot | \mathbf{M}_\cdot)$  of a Stixel column  $\mathbf{S}_\cdot$  given the measurements  $\mathbf{M}_\cdot$  within the column based on ideas presented by Pfeiffer and Franke, 2011b. These measurements usually consist of a dense disparity map  $\mathbf{D}_\cdot$ , the color image  $\mathbf{I}_\cdot$ , and pixel-level semantic label scores  $\mathbf{L}_\cdot$ . However, not all of these channels are required simultaneously and we will discuss a differing constellation in Section 5.6. We describe the posterior by means of a graphical model to allow for an intuitive and clear formalism, a structured description of statistical independence, and a visualization

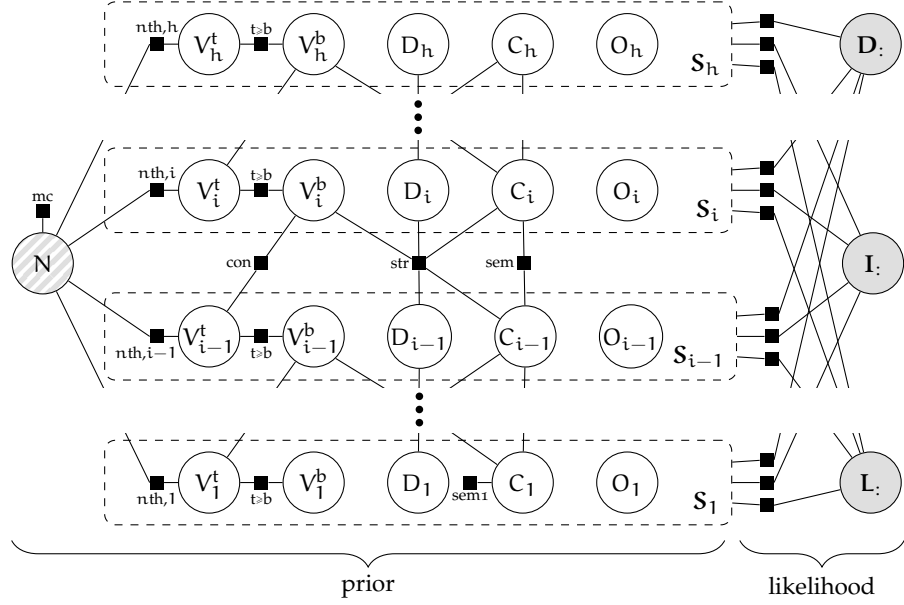


Figure 5.2: The Stixel world as a factor graph that depicts the factorization of the posterior distribution, c.f. Equation (5.1). Each Stixel  $S_i$  (dashed boxes) stands for the five random variables  $V_i^b, V_i^t, C_i, D_i, O_i$  (circles) denoting the Stixel's vertical extent from bottom to top row, its semantic class label, its 3D position, and its color attribute. The hatched node on the left denotes the random variable  $N$  describing the number of Stixels  $n$  that constitute the final segmentation. Black squares denote factors of the posterior and are labeled according to the descriptions given in the text. The prior distribution factorizes according to the left part, whereas the right part describes the measurement likelihood. The circles  $D:$ ,  $I:$ , and  $L:$  denote the measurements, i.e. a column of the disparity map, the color image, and the pixel-level semantic label scores, respectively. If all measurements are observed (indicated by gray shading) and if the number of Stixels  $N$  is thought to be fixed (indicated by gray hatched shading), the graph is chain-structured on the Stixel level. This property is exploited for inference via dynamic programming (see Section 5.4).

via a factor graph, c.f. Figure 5.2. The graph provides two high-level factors. The first rates the *likelihood* of a measurement given a candidate Stixel column and is denoted as  $\tilde{P}(\mathbf{M}_i | \mathbf{S}_i)$ . The second factor models our a-priori knowledge about street scenes, i.e. the *prior*  $\tilde{P}(\mathbf{S}_i)$ . Note that both, likelihood and prior are unnormalized probability mass functions. Together with the normalizing partition function  $Z$ , the posterior distribution is defined as

$$P(\mathbf{S}_i | \mathbf{M}_i) = \frac{1}{Z} \tilde{P}(\mathbf{M}_i | \mathbf{S}_i) \tilde{P}(\mathbf{S}_i) . \quad (5.1)$$

For the ease of notation, we switch to the log-domain and obtain

$$P(\mathbf{S}_i = \mathbf{s}_i | \mathbf{M}_i = \mathbf{m}_i) = \frac{1}{Z(\mathbf{m}_i)} e^{-E(\mathbf{s}_i, \mathbf{m}_i)} , \quad (5.2)$$

where  $E(\cdot)$  is the energy function, defined as

$$E(\mathbf{s}_i, \mathbf{m}_i) = \Phi(\mathbf{s}_i, \mathbf{m}_i) + \Psi(\mathbf{s}_i) . \quad (5.3)$$

The function  $\Phi(\cdot)$  represents the likelihood and  $\Psi(\cdot)$  the prior.

In order to mathematically define the posterior energy function, we need a fixed number of random variables describing the segmentation  $\mathbf{S}_i$ . This is accomplished by splitting the column  $\mathbf{S}_i$  into as many individual Stixels  $\mathbf{S}_i$  as maximally possible, i.e.  $i \in \{1 \dots h\}$ , and into an additional random variable  $N \in \{1 \dots h\}$  denoting the number of Stixels that constitute the final segmentation. For a certain value  $n$ , we set all factors connected to a Stixel  $\mathbf{S}_i$  with  $i > n$  to zero energy. Thus, these factors do not influence the segmentation obtained. For ease of notation, we continue by assuming  $i \leq n$  and drop the dependency of the factors on  $N$ . We revisit these simplifications and discuss their impact during inference later in Section 5.4.

Besides the random variable  $N$ , the column segmentation  $\mathbf{S}_i$  consists of  $h$  Stixels  $\mathbf{S}_i$ , where  $\mathbf{S}_1$  is the lowest Stixel and  $\mathbf{S}_h$  the highest. A Stixel  $\mathbf{S}_i$  is in turn split into the five random variables  $V_i^b$ ,  $V_i^t$ ,  $C_i$ ,  $O_i$ , and  $D_i$ . The first two denote the vertical extent from bottom to top row. The variable  $C_i$  represents the Stixel's semantic class label (and in turn the structural class via the label hierarchy),  $O_i$  is its color attribute, and  $D_i$  parameterizes the disparity model. Note that a vertical segment at constant distance maps to a constant disparity, while a support segment at constant height maps to a constant disparity offset relative to the ground plane. Hence, both, the distance and the height are captured by a single random variable  $D_i$ .

### 5.3.1 Prior

The prior  $\Psi(\mathbf{s}_i)$  from Equation (5.3) captures prior knowledge on the scene structure independent from any measurements. Such a model serves as regularization of the segmentation and becomes especially important in challenging scenarios such as bad weather or

night scenes where the measurements are noisy. The prior term factorizes as

$$\Psi(\mathbf{s};) = \Psi_{\text{mc}}(\mathbf{n}) + \sum_{i=1}^h \sum_{id} \Psi_{id}(\mathbf{s}_i, \mathbf{s}_{i-1}, \mathbf{n}) , \quad (5.4)$$

where *id* stands for the name of the factors corresponding to their labels in Figure 5.2. Note that not all factors actually depend on all variables  $\mathbf{s}_i$ ,  $\mathbf{s}_{i-1}$ , or  $\mathbf{n}$ . In the following, we define and explain the individual factors by grouping them regarding their functionality, i.e. model complexity, segmentation consistency, structural priors, and semantic priors.

**MODEL COMPLEXITY** The model complexity prior  $\Psi_{\text{mc}}$  is the main regularization term and controls the compromise between compactness and robustness versus fine granularity and accuracy. The factor is defined as

$$\Psi_{\text{mc}}(\mathbf{n}) = \beta_{\text{mc}} \mathbf{n} . \quad (5.5)$$

The higher the parameter  $\beta_{\text{mc}}$  is chosen, the fewer Stixels are obtained, hence the segmentation becomes more compact.

**SEGMENTATION CONSISTENCY** In order to obtain a consistent segmentation, we define hard constraints to satisfy that segments are non-overlapping, connected, and extend over the whole image. This implies that the first Stixel must begin in image row 1 (bottom row) and the last Stixel must end in row  $h$  (top row), i.e.

$$\Psi_{1\text{st}}(v_1^b) = \begin{cases} 0 & \text{if } v_1^b = 1 \\ \infty & \text{otherwise} , \end{cases} \quad (5.6)$$

$$\Psi_{\text{nth},i}(\mathbf{n}, v_i^t) = \begin{cases} \infty & \text{if } \mathbf{n} = i \text{ and } v_i^t \neq h \\ 0 & \text{otherwise} . \end{cases} \quad (5.7)$$

Further, a Stixel's top row must be above the bottom row and consecutive Stixels must be connected, i.e.

$$\Psi_{t \geq b}(v_i^b, v_i^t) = \begin{cases} 0 & \text{if } v_i^b \leq v_i^t \\ \infty & \text{otherwise} , \end{cases} \quad (5.8)$$

$$\Psi_{\text{con}}(v_i^b, v_{i-1}^t) = \begin{cases} 0 & \text{if } v_i^b = v_{i-1}^t + 1 \\ \infty & \text{otherwise} . \end{cases} \quad (5.9)$$

In Section 5.4, we will show that these deterministic constraints can be exploited to significantly reduce the computational effort.

**STRUCTURAL PRIORS** Road scenes have a typical 3D layout in terms of the structural classes support, vertical, and sky. This lay-

out is modeled by  $\Psi_{\text{str}}$ , which is in turn comprised of two factors, both responsible for an individual effect, i.e.

$$\Psi_{\text{str}}(c_i, c_{i-1}, d_i, d_{i-1}, v_i^b) = \Psi_{\text{grav}} + \Psi_{\text{do}}. \quad (5.10)$$

The gravity component  $\Psi_{\text{grav}}$  is only non-zero for  $c_i \in \mathcal{V}$  and  $c_{i-1} \in \mathcal{S}$  and models that in 3D a vertical segment usually stands on the preceding support surface. Consequently, the vertical segment's disparity  $d_i$  and the disparity of the support surface must coincide in row  $v_i^b$ . The latter disparity is denoted by the function  $d_s(v_i^b, d_{i-1})$  and their difference as  $\Delta_d = d_i - d_s(v_i^b, d_{i-1})$ . Then,  $\Psi_{\text{grav}}$  is defined as

$$\Psi_{\text{grav}} = \begin{cases} \alpha_{\text{grav}}^- + \beta_{\text{grav}}^- \Delta_d & \text{if } \Delta_d < 0 \\ \alpha_{\text{grav}}^+ + \beta_{\text{grav}}^+ \Delta_d & \text{if } \Delta_d > 0 \\ 0 & \text{otherwise} \end{cases} \quad (5.11)$$

The second term  $\Psi_{\text{do}}$  rates the depth ordering of vertical segments. Usually, an object that is located on top of another object in the image is behind the other one in the 3D scene, i.e. the top disparity is smaller than the bottom one. Therefore, we define

$$\Psi_{\text{do}} = \begin{cases} \alpha_{\text{do}} + \beta_{\text{do}} (d_i - d_{i-1}) & \text{if } c_i \in \mathcal{V}, c_{i-1} \in \mathcal{V}, d_i > d_{i-1} \\ 0 & \text{otherwise} \end{cases} \quad (5.12)$$

**SEMANTIC PRIORS** The last group of prior factors is responsible for the a-priori knowledge regarding the semantic structure of road scenes. We define the factor as

$$\Psi_{\text{sem}}(c_i, c_{i-1}) = \gamma_{c_i} + \gamma_{c_i, c_{i-1}}. \quad (5.13)$$

The first term  $\gamma_{c_i}$  describes the a-priori class probability for a certain class  $c_i$ , i.e. the higher the values are, the less likely are Stixels with that class label. The latter term  $\gamma_{c_i, c_{i-1}}$  is defined via a two-dimensional transition matrix  $\gamma_{c_i, c_{i-1}}$  for all combinations of classes. Individual entries in this matrix model expectations on relative class locations, e.g. a support Stixel such as road above a vertical Stixel such as car might be rated less likely than vice versa. Note that we capture only first order relations to allow for efficient inference. Finally, we define  $\Psi_{\text{sem1}}(c_1)$  analogously for the first Stixel's class, e.g. a first Stixel with a support class such as road might be more likely than with a vertical class such as infrastructure or sky.

### 5.3.2 Data likelihood

The data likelihood from Equation (5.3) integrates the information from our input modalities (disparity map  $\mathbf{D}$ ., color image  $\mathbf{I}$ ., and semantic label scores  $\mathbf{L}$ .) and rates their compatibility with a candidate



Stixel column. We rely on multiple input modalities mainly for redundancy and in order to combine their individual strengths. The depth information allows to separate support from vertical segments as well as multiple stacked objects at different distances. Furthermore, these data terms are in principle independent of the actual object type and therefore generalize very well to varying scene types and content. The semantic channel in turn is based on classifiers for a specific set of classes. For these classes it shows excellent recognition performance, but might fail for novel semantic classes. Further, such a classifier typically does not separate multiple neighboring instances of the same semantic class, e.g. multiple cars behind each other, while the depth information supports their segmentation. Our last channel, the color image itself, is a comparably weak source of information, but nevertheless helps to localize precise object boundaries and is useful whenever an over-segmentation that captures all image boundaries is desirable.

The likelihood term factorizes as

$$\Phi(\mathbf{s}_:, \mathbf{m}_:) = \sum_{i=1}^h \sum_{v=v_i^b}^{v_i^t} \Phi_D(\mathbf{s}_i, d_v, v) + \Phi_I(\mathbf{s}_i, i_v) + \Phi_L(\mathbf{s}_i, l_v) . \quad (5.14)$$

Note that we sum over the maximum number of Stixels  $h$ , but as described above, all factors for  $i > n$  are set to zero. Further, for a given Stixel segmentation  $\mathbf{s}_:$ , we model the data likelihoods to be independent across pixels and therefore their contribution decomposes over the rows  $v$ . In the following, we describe the contributions of the individual modalities to a Stixel  $\mathbf{s}_i$ .

**DEPTH** The depth likelihood terms are designed according to our world model consisting of supporting and vertical planar surfaces. Since the width  $w$  of a Stixel is rather small, we can neglect the influence of slanted surfaces. Instead, these are represented, with some discretization, via neighboring Stixels at varying depths. In doing so, the 3D orientation of a Stixel is sufficiently described by its structural class, i.e. support or vertical. Accordingly, the 3D position of a Stixel is parametrized by a single variable  $D_v$ , paired with its 2D position in the image. This variable is the Stixel's constant disparity for a vertical segment and a constant disparity offset relative to the ground plane for a support segment, c.f. Figure 5.3.

We use depth measurements in the form of dense disparity maps, where each pixel has an associated disparity value or is flagged as invalid, i.e.  $d_v \in \{0 \dots d_{\max}, d_{\text{inv}}\}$ . The subscript  $v$  denotes the row index within the considered column. The depth likelihood term  $\Phi_D(\mathbf{s}_i, d_v, v)$  is derived from a probabilistic, generative measurement model  $P_v(D_v = d_v \mid \mathbf{S}_i = \mathbf{s}_i)$  according to

$$\Phi_D(\mathbf{s}_i, d_v, v) = -\delta_D(c_i) \log(P_v(D_v = d_v \mid \mathbf{S}_i = \mathbf{s}_i)) . \quad (5.15)$$



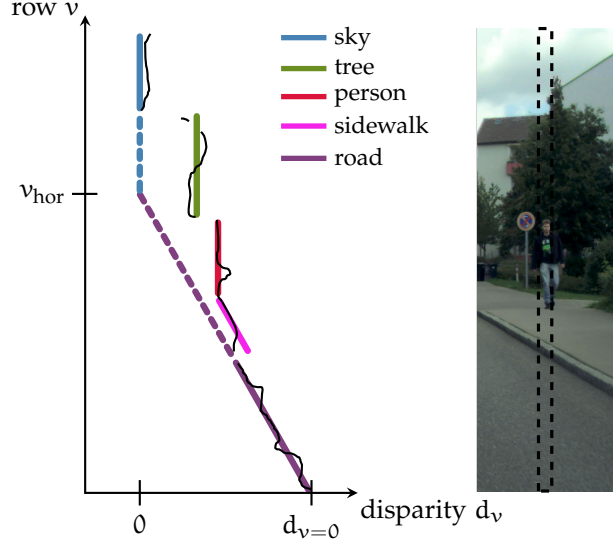


Figure 5.3: Disparity measurements (black lines) and resulting Stixel segmentation (colored lines) for a typical image column (right). The purple dashed line symbolizes the linear ideal disparity measurements along a planar ground surface that support segments such as road or sidewalk are parallel to. Obstacles form vertical Stixels, e.g. person or tree. Sky is modeled with a disparity value of zero. Adapted from Pfeiffer, 2011.

The term incorporates class-specific weights  $\delta_D(c_i)$  that allow to learn the relevance of the depth information for a certain class. Let  $p_{\text{val}}$  be the prior probability of a valid disparity measurement. Then, we obtain

$$P_v(D_v | \mathbf{S}_i) = \begin{cases} p_{\text{val}} P_{v,\text{val}}(D_v | \mathbf{S}_i) & \text{if } d_v \neq d_{\text{inv}} \\ (1 - p_{\text{val}}) & \text{otherwise,} \end{cases} \quad (5.16)$$

where  $P_{v,\text{val}}(D_v | \mathbf{S}_i)$  denotes the measurement model of valid measurements only and is defined as

$$P_{v,\text{val}}(D_v | \mathbf{S}_i) = \frac{p_{\text{out}}}{Z_U} + \frac{1 - p_{\text{out}}}{Z_G(\mathbf{S}_i)} e^{-\frac{1}{2} \left( \frac{d_v - \mu(\mathbf{S}_i, v)}{\sigma(\mathbf{S}_i)} \right)^2}. \quad (5.17)$$

This distribution is a mixture of a uniform and a Gaussian distribution and defines the sensor model. While the Gaussian captures typical disparity measurement noise, the uniform distribution increases the robustness against outliers and is weighted with the coefficient  $p_{\text{out}}$ . The Gaussian is centered at the disparity value  $\mu(\mathbf{S}_i, v)$  of the Stixel  $\mathbf{s}_i$ , which is constant for vertical Stixels, i.e.  $\mu(\mathbf{S}_i, v) = d_i$ , and depends on row  $v$  for support Stixels, c.f. Figure 5.3. The standard deviation  $\sigma$  captures the noise properties of the stereo algorithm and is chosen depending on the class  $c_i$ , e.g. for class sky the noise is expected to be higher than for the other classes due to missing texture as needed by stereo matching algorithms. The parameters  $p_{\text{val}}$ ,  $p_{\text{out}}$ ,

and  $\sigma$  are either chosen a-priori, can be obtained by estimating confidences in the stereo matching algorithm as shown by Pfeiffer et al., 2013, or can be chosen based on empirical measurements as described in Cordts et al., 2017b. The terms  $Z_G(s_i)$  and  $Z_U$  normalize the two distributions.

**COLOR IMAGE** Common superpixel algorithms such as Simple Linear Iterative Clustering (SLIC, Achanta et al., 2012) work by grouping adjacent pixels of similar color. We follow this idea by favoring Stixels with a small squared deviation in LAB color space from their color attribute  $O_i$ . Thus, the color likelihood  $\Phi_I(s_i, i_v)$  in Equation (5.14) is defined as

$$\Phi_I(s_i, i_v) = \delta_I(c_i) \|i_v - o_i\|_2^2. \quad (5.18)$$

Since some classes might satisfy a constant color assumption better than others, e.g. sky vs. object, each class is associated with an individual weight  $\delta_I$ . Note that it is straightforward to extend this likelihood term to more sophisticated color or texture models. In this work, we opt for semantic label scores to incorporate appearance information.

**SEMANTIC LABEL SCORES** The driving force in terms of semantic scene information is provided by a pixel-level labeling system that delivers normalized semantic scores  $l_v(c_i)$  with  $\sum_{c_i} l_v(c_i) = 1$  for all considered classes  $c_i$  at all pixels  $v$ . This input channel not only separates the structural classes into their subordinate semantic classes, but also guides the segmentation by leveraging appearance information. The semantic scores yield the likelihood term

$$\Phi_L(s_i, l_v) = -\delta_L(c_i) \log(l_v(c_i)) \quad (5.19)$$

in Equation (5.14). Again, we use class-specific weights  $\delta_L(c_i)$ .

#### 5.4 INFERENCE

We perform inference by finding the maximum-a-posteriori (MAP) solution by maximizing Equation (5.1), or equivalently by minimizing the energy function in Equation (5.3). One motivation for this is that as opposed to a maximum marginal estimate, the obtained segmentation is consistent in terms of the constraints described in Section 5.3.1. We describe the inference algorithm in three stages: first using a naive solution, then by exploiting algorithmic simplifications, and eventually by using slight approximations to further reduce the computational effort.

### 5.4.1 Dynamic programming

If we treat the given measurements as implicit parameters and combine Equations (5.3), (5.4) and (5.14), the optimization problem has the structure

$$\begin{aligned} \mathbf{s}_\dagger^* = \operatorname{argmin}_{n, \mathbf{s}_1 \dots \mathbf{s}_h} & \Psi_{\text{mc}}(n) + \Phi_1(\mathbf{s}_1, n) + \Psi_1(\mathbf{s}_1, n) \\ & + \sum_{i=2}^h \Phi_i(\mathbf{s}_i, n) + \Psi_i(\mathbf{s}_i, \mathbf{s}_{i-1}, n) , \end{aligned} \quad (5.20)$$

where all prior and likelihood factors of a single Stixel  $i$  or a pair of neighboring Stixels are grouped together. A straightforward way to solve Equation (5.20) is to iterate over all possible numbers of Stixels  $n$  and solve

$$\begin{aligned} c_n^* &= \Psi_{\text{mc}}(n) \\ &+ \min_{\mathbf{s}_1 \dots \mathbf{s}_h} \Phi_{1,n}(\mathbf{s}_1) + \Psi_{1,n}(\mathbf{s}_1) + \sum_{i=2}^h \Phi_{i,n}(\mathbf{s}_i) + \Psi_{i,n}(\mathbf{s}_i, \mathbf{s}_{i-1}) \end{aligned} \quad (5.21)$$

for each fixed  $n$ . The minimum value of all  $c_n^*$  determines  $n^*$  and the minimizing segmentation  $\mathbf{s}_\dagger^*$  is the optimal segmentation of the current column. Next, for a fixed  $n$ , we exploit that  $\Phi_{i,n} = \Psi_{i,n} = 0$  for  $i > n$  and that the factor  $\Psi_{\text{nth},i}$  reduces to the constraint  $v_n^t = h$ . Besides that, neither  $\Phi_{i,n}$  nor  $\Psi_{i,n}$  depend on  $n$  or  $i$  (except for  $i = 1$ ) and it holds that

$$c_n^* = \Psi_{\text{mc}}(n) + \min_{\substack{\mathbf{s}_1 \dots \mathbf{s}_n \\ v_n^t = h}} \Phi(\mathbf{s}_1) + \Psi_1(\mathbf{s}_1) + \sum_{i=2}^n \Phi(\mathbf{s}_i) + \Psi(\mathbf{s}_i, \mathbf{s}_{i-1}) . \quad (5.22)$$

Due to the first-order Markov property on Stixel super-nodes, c.f. Figure 5.2, Equation (5.22) can be solved via the Viterbi algorithm, i.e. Dynamic Programming (DP), by reformulation as

$$\begin{aligned} c_n^* &= \Psi_{\text{mc}}(n) + \min_{\mathbf{s}_n, v_n^t = h} \left( \Phi(\mathbf{s}_n) \right. \\ &\quad + \min_{\mathbf{s}_{n-1}} \left( \Phi(\mathbf{s}_{n-1}) + \Psi(\mathbf{s}_n, \mathbf{s}_{n-1}) \right. \\ &\quad \quad \vdots \\ &\quad \quad \left. \left. + \min_{\mathbf{s}_1} \left( \Phi(\mathbf{s}_1) + \Psi(\mathbf{s}_2, \mathbf{s}_1) + \Psi_1(\mathbf{s}_1) \right) \dots \right) \right) . \end{aligned} \quad (5.23)$$

The number of possible states of a Stixel  $\mathbf{S}_i$  is

$$|\mathbf{S}_i| = |V^t| |V^b| |C| |O| |D| = h^2 |C| |O| |D| , \quad (5.24)$$

and hence we obtain a run time of  $\mathcal{O}(N |S_i|^2) = \mathcal{O}(Nh^4 |C|^2 |O|^2 |D|^2)$  for each value of  $n$ . Since inference is run for  $n \in \{1 \dots h\}$ , the overall run time is  $\mathcal{O}(h^6 |C|^2 |O|^2 |D|^2)$ . This assumes that all prior and likelihood factors can be computed in constant time. In case of the priors, this property becomes evident from the definitions in Section 5.3.1. For the data likelihoods, constant run time is achieved by leveraging integral tables within each image column. For the disparity data term, we apply the approximations described in Pfeiffer, 2011 to compute the integral table of the disparity measurements. Overall, the asymptotic run time of such pre-computations is small compared to the inference via the Viterbi algorithm and can thus be neglected.

#### 5.4.2 Algorithmic simplification

The run time can be significantly reduced by exploiting the structure of the optimization problem in Equation (5.23). All intermediate problems neither depend on the number of Stixels  $n$  nor on the actual Stixel index  $i$ , except for  $i = 1$ . Further, the model complexity factor  $\Psi_{mc}(n)$  is linear in the number of Stixels, c.f. Equation (5.5), and can thus be transformed into a constant unary term for each Stixel. Therefore, inference can be performed jointly for all values of  $n$  and the overall run time reduces to  $\mathcal{O}(|S_i|^2) = \mathcal{O}(h^4 |C|^2 |O|^2 |D|^2)$ .

Further improvement is obtained by exploiting the deterministic constraints on the segmentation consistency, c.f. Section 5.3.1. The random variable  $V_i^b$  can be substituted with  $V_{i-1}^t + 1$  for  $i > 1$  and with 1 for  $i = 1$ , c.f. the factors  $\Psi_{con}$  and  $\Psi_{1st}$  in Equations (5.6) and (5.9). As shown in Figure 5.2, there is no connection between  $V_i^b$  and any random variable from Stixel  $S_{i+1}$ . Thus, the substitution does not add a second-order dependency and an inference via the Viterbi algorithm is still possible. However, the number of states is reduced to

$$|S_i| = |V^t| |C| |O| |D| = h |C| |O| |D| , \quad (5.25)$$

and the overall run time becomes  $\mathcal{O}(h^2 |C|^2 |O|^2 |D|^2)$ .

An inspection of all factors defined in Section 5.3 unveils that no pairwise factor depends on the Stixel's color attribute  $O_i$ . Instead, only the image data likelihood in Equation (5.18) depends on  $O_i$  and its minimization is solved analytically, i.e.  $o_i^*$  is the mean image within the Stixel  $s_i$ . Both,  $o_i^*$  and the value of Equation (5.18), can be computed in constant time using integral tables. The run time is therefore reduced to  $\mathcal{O}(h^2 |C|^2 |D|^2)$ .

Let us now discuss the role of  $C_i$ , i.e. the Stixel's semantic class and indirectly its structural class<sup>1</sup>. The structural prior in Equation (5.10) depends only on the structural class, but not on the semantic class, while the semantic prior in Equation (5.13) can in principle model transitions between semantic classes. However, in practice it is sufficient to restrict this prior to structural classes only and add exceptions for a few select classes depending on the actual application, e.g. Section 5.6. The same holds for the weights of the data likelihoods, i.e.  $\delta_D(c_i)$ ,  $\delta_I(c_i)$ ,  $\delta_L(c_i)$ . Since there is a fixed number of three structural classes referring to our world model, the evaluation of all pairwise terms is constant with respect to the number of classes. Only the data likelihood in Equation (5.19) depends on  $|C|$ , yielding a run time of  $\mathcal{O}(h^2 |C| |D|^2)$ .

### 5.4.3 Approximations

To further reduce the inference effort, Pfeiffer, 2011 proposed to not infer the variables  $D_i$  that describe the Stixels' 3D positions. Instead, the value is computed from the given disparity map depending on the Stixel's extent and structural class label by averaging the disparities within a vertical Stixel respectively the disparity offsets within a support Stixel. In doing so, the isolated disparity data likelihood terms are minimized individually, but the global minimum including the priors is not found exactly. Note that the Stixel's 3D position still depends on its extent, which in turn is inferred by solving the global minimization problem. The average disparities can be computed in constant time using integral tables and the overall run time reduces to  $\mathcal{O}(h^2 |C|)$ , which is a significant improvement compared to the initial run time of  $\mathcal{O}(h^6 |C|^2 |O|^2 |D|^2)$ .

A different interpretation of the inference problem is to find the shortest path in the Directed Acyclic Graph (DAG) in Figure 5.4. The edge weights in that graph are defined according to Equation (5.22). For an edge between two Stixels  $s_l$  (left) and  $s_r$  (right), the weight is  $\Phi(s_l) + \Psi(s_r, s_l)$ . Between the source and a Stixel  $s_r$ , the weight is  $\Psi_1(s_r)$  and between a Stixel  $s_l$  and the sink we obtain  $\Phi(s_l)$ . Note that in all three cases, the model complexity factor is decomposed over these weights, yielding an additional term  $\beta_{mc}$ . Solving the shortest path problem is bound by the number of edges  $\mathcal{O}(h^2)$  and finding the optimal semantic class for each edge in  $\mathcal{O}(|C|)$ , yielding  $\mathcal{O}(h^2 |C|)$ .

<sup>1</sup> The described simplification was originally proposed by Lukas Schneider (L. Schneider et al., 2016) and is here embedded into the formalism provided by the graphical model.

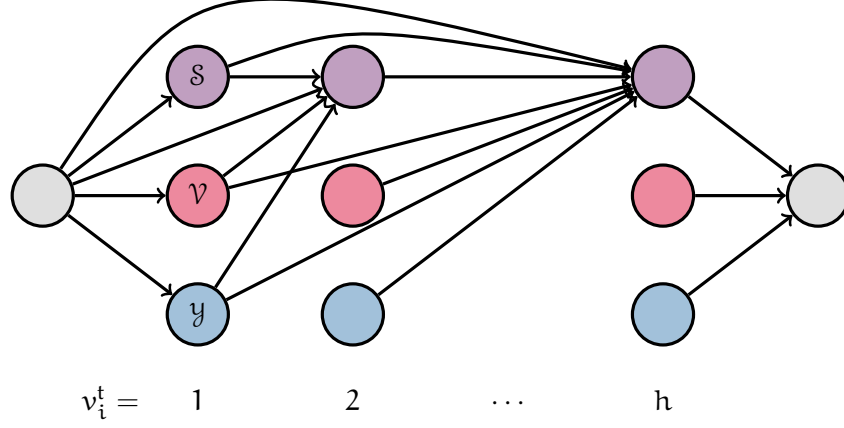


Figure 5.4: Stixel inference algorithm as a shortest path problem, where the Stixel segmentation is obtained by the colored nodes along the shortest path from the source (left gray node) to the sink (right gray node). The color of the circles denotes the Stixel’s structural class, i.e. support (purple), vertical (red), and sky (blue). The horizontal position of the nodes is the Stixel’s top row  $v_i^t$ . The graph is a Directed Acyclic Graph (DAG), where only the incoming edges of support nodes are shown.

## 5.5 PARAMETER LEARNING

The Stixel segmentation is obtained by minimizing the energy function as introduced in Section 5.3, which is in principle controlled by two groups of parameters. The first group holds the parameters of the generative disparity model, which accounts for the planar Stixel assumption as well as the measurement uncertainty of depth estimates, i.e.  $p_{\text{val}}$ ,  $p_{\text{out}}$ , and  $\sigma$  in Equation (5.16) and Equation (5.17). In order to derive these parameters from data or stereo confidences, the reader is referred to Cordts et al., 2017b and Pfeiffer et al., 2013, respectively.

The energy function is linear in all remaining parameters forming the second group of parameters, i.e. model complexity ( $\beta_{\text{mc}}$ ), structural priors ( $\alpha_{\text{grav}}^-, \beta_{\text{grav}}^-, \alpha_{\text{grav}}^+, \beta_{\text{grav}}^+, \alpha_{\text{do}}, \beta_{\text{do}}$ ), semantic priors ( $\gamma_{c_i}, \gamma_{c_i, c_{i-1}}$ ) and the weights between the data likelihoods ( $\delta_{\text{D}}(c_i), \delta_{\text{I}}(c_i), \delta_{\text{L}}(c_i)$ ). As tuning these parameters manually can be a tedious task, we investigate automatic parameter learning by means of a Structured Support Vector Machine (S-SVM) with margin-rescaled Hinge loss (Nowozin and Lampert, 2011). To do so, we generate ground truth Stixels using the Cityscapes dataset, c.f. Chapter 4. We cut the instance-level annotations into individual columns and assign the corresponding ground truth class label to every instance segment. Further, we approximate ground truth 3D models according to the planar assumptions of support and vertical Stixels. To this end, we base on the median SGM depth measurements (Gehrig et al., 2009;

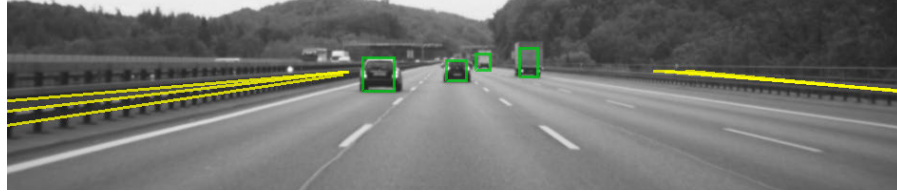
Hirschmüller, 2008) within a Stixel assuming that this approximation is sufficient to estimate the parameters we are interested in.

Next, we define a loss that compares a ground truth Stixel segmentation with an inferred one. For each inferred Stixel, we determine the maximum overlapping ground truth Stixel and compute the squared deviations between the top and bottom rows that we add to the loss. We punish under-segmentations by two orders of magnitude stronger than over-segmentations since under-segmentation might result in an undetected obstacle, which is the worst-case scenario for an autonomous vehicle. The induced loss of both cases is truncated after a significant deviation between inferred and ground truth Stixel, considering the cut as missed. In addition to segmentation errors, we add the number of pixels with erroneous structural class in each column as additional loss. In doing so, the loss decomposes over the inferred Stixels and can be treated as a fourth data term, hence the inference during training is tractable using the same method and approximations as in Section 5.4. While the sketched structured learning approach works in principle and converges to a meaningful parameter set, we found that the obtained results are comparable, but never surpass manual tuning that is guided by empirical observations of street scenes. We experimented with different loss variants, but in all cases manually tuned parameters were still slightly better, even in terms of the loss used for training. We attribute this effect to the convex relaxation of the Hinge loss, which in our case may not be tight enough. Hence, parameter learning for the Stixel model yields a valid starting point, but further manual guidance by physical evidence helps to increase robustness and generalization abilities.

## 5.6 OBJECT-LEVEL KNOWLEDGE

In Section 5.3.2, we proposed to leverage semantic information from a pixel-level semantic labeling system as data term for the Stixel generation. Since such a system shows excellent recognition performance for complex urban scenes with various types of objects in often highly occluded constellations, c.f. Chapter 4, the main focus of this dissertation is on such pixel-level semantic parsing. However, bounding box object detectors (Felzenszwalb et al., 2010; Girshick, 2015; W. Liu et al., 2016; Redmon et al., 2016; Viola and Jones, 2004) also show excellent performance: such detectors were successfully used for autonomous driving systems (Franke et al., 2013) and are already available in series production vehicles, e.g. a pedestrian detector based on Enzweiler and Gavrila, 2009 in Mercedes-Benz cars. Besides bounding box detectors, object-level knowledge can also stem from instance-level segmentation, c.f. Section 4.4 or simple line detectors for guard rails, e.g. Scharwächter et al., 2014b.





(a) Two different types of object-level knowledge. Vehicles are represented by bounding boxes whereas guard rails are described by 2D lines delimiting their top extent.



(b) Resulting top/bottom cut energy maps for bounding box and line detections.

Figure 5.5: Examples of object-level knowledge that we incorporate into our Stixel model. Detections (top) are transformed into top and bottom cut energy maps (bottom) that we leverage as data terms during Stixel inference.

In this section, we will briefly discuss a variant of the Stixel model that leverages such object-level knowledge. In doing so, we also underline the flexibility and generality of our CRF-based Stixel model. For details on the extension and an experimental analysis, the reader is referred to Cordts et al., 2014.

#### 5.6.1 Extended Stixel model

From a high-level perspective, both object detectors and pixel-level semantic labeling systems deliver semantic information about the scene. Thus, one could imagine to treat both modules analogously without a need for further modifications. However, taking a closer look, we notice substantial differences in the representational power of the two components. While pixel-level labeling delivers a class segmentation, it does not separate individual object instances from each other. In contrast, the object-level knowledge that we would like to exploit in this discussion provides exactly this type of information. In turn, bounding boxes do not aim for an accurate segmentation and in case of a line detector we might only have information about the end of an object, c.f. Figure 5.5a.

In order to cope with these different types of object-level information, we introduce a new data likelihood factor as well as an object height prior, c.f. Figure 5.6. Further, we discuss the usage of specialized semantic priors. Note that for simplicity, we base this section on depth and object-level data terms only.



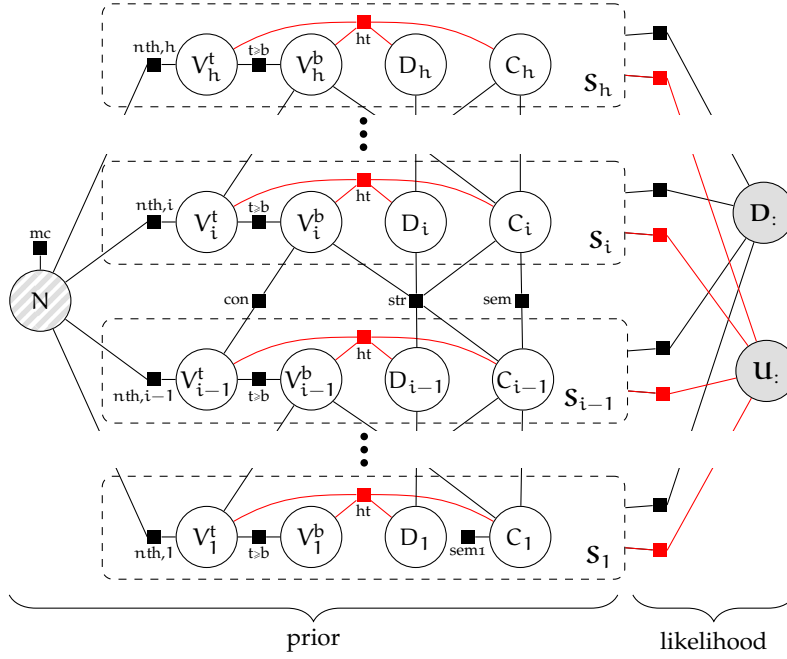


Figure 5.6: The Stixel world as a factor graph that depicts the factorization of the posterior distribution, c.f. Equation (5.1). The formulation in Section 5.3 and Figure 5.2 is extended by two factors (depicted in red) that are specialized for object-level knowledge, e.g. stemming from bounding box object detectors. For simplicity, we removed the pixel-level semantic data term as well as the image channel and the corresponding attributes.

**OBJECT-LEVEL LIKELIHOOD** As discussed above, we desire to incorporate various types of object-level information into the Stixel model. To achieve such flexibility, we introduce so called top/bottom cut energy maps as a proxy representation that unifies these various types into a common scheme, c.f. Figure 5.5b. In other words, we assume that we have a model describing for all pixels and all object types a probability score, or equivalently energy, of being a bottom or top point of the object. The model takes into account the reliability of the source of information, its importance for the Stixel generation, and also a possible uncertainty in the precise location. Such a mapping from a given contour to actual bottom and top point energies could either be obtained from training data or by blurring the contour. In image regions without object detections or for semantic classes without matching detector, we set the maps to zero energy or positive values depending on the typical miss rates of the object detector.

Let  $u^t(v, c_i)$  be the energy for row  $v$  being the top row of a Stixel with semantic class  $c_i$  and let  $u^b(v, c_i)$  be defined analogously for the bottom row. Then, the data likelihood factor of a candidate Stixel  $s_i$  is defined as

$$\Phi_U(s_i, \mathbf{u}^t, \mathbf{u}^b) = u^t(v_i^t, c_i) + u^b(v_i^b, c_i) \quad (5.26)$$

Note that this likelihood factor has a slightly different structure than those defined in Section 5.3.2. It does not decompose over pixels such that it can capture knowledge about object instances rather than individual pixels. Nevertheless, for inference the changed structure does not impose any asymptotic run time changes as the factor can be evaluated in constant time for each Stixel candidate  $s_i$ .

**HEIGHT PRIOR** Since object-level information enables a reasoning about individual object instances, we can introduce a height prior that models the typical heights of these instances. Such a prior helps especially, when the object-level information is weak, e.g. the guard rail line detector shown in Figure 5.5 that delivers only the top location of guard rails: without a height prior, all Stixels with a matching top row would become guard rail. Naturally, such a prior makes more sense when occlusions are rare, e.g. in highway scenarios as considered in Cordts et al., 2014.

In order to model the height of an object as represented by a Stixel, we introduce the factor  $\Psi_{ht}$  for *vertical* Stixels. Using the Stixel's 2D extent, its distance and known camera parameters, we can compute the height in the 3D world. Depending on the Stixel's semantic class, implausible heights are then rated down. Again, this factor can be evaluated in constant time and the asymptotic run time does not change.

**SPECIALIZED SEMANTIC PRIOR** In the 3D world, the semantic classes that we consider in this section, i.e. *vehicle* and *guard rail*, are typically located on the ground. When occlusions are rare, e.g. in highway scenarios as shown in Figure 5.5, such objects are typically also on top of ground regions in the 2D image domain. Thus, the semantic prior  $\Psi_{\text{sem}}(c_i, c_{i-1})$  as introduced in Equation (5.13) can be used to capture these expectations by modifying  $\gamma_{c_i, c_{i-1}}$  appropriately. Note however, that the run time analysis in Section 5.4.2 is based on modeling relations between structural classes only, while such modifications require semantic relations. In practice, if only a few semantic classes are explicitly modeled, the inference is still tractable. If the modifications become hard constraints on the relative class locations, e.g. no vehicle or guard rail below the ground surface allowed (Cordts et al., 2014), the constraints can be exploited to mitigate the impacts on run time.

## 5.7 EXPERIMENTS<sup>2</sup>

In this section, we evaluate the Stixel model to analyze the influence of our input cues (Section 5.7.4) and the impact of our approximations for efficient inference (Section 5.7.5). To this end, we inspect the model’s accuracies in terms of depth and semantic labeling estimation. For an extensive analysis of the underlying environment model and the effect of important model parameters the reader is referred to Cordts et al., 2017b. It turns out that the Stixel segmentation can indeed achieve high accuracies paired with strong compression rates. Prior information helps in particular when facing challenging weather conditions, where robustness and regularization are beneficial. Exemplary results of our Semantic Stixel model on Cityscapes can be found in Figure 5.7.

For our experiments, we use SGM (Gehrig et al., 2009; Hirschmüller, 2008) to obtain dense disparity maps as input channel. For semantic label estimates, we rely on FCN (Shelhamer et al., 2017) using GoogLeNet (Szegedy et al., 2015) as underlying neural network. The used model is an early variant of those in Section 4.3.5 and has worse performance. However, for the analyses in this section, the actual source of the input channels is irrelevant, the absolute performance of the Semantic Stixels scales directly with the quality of the inputs and the observed relative performance differences are expected to be widely independent of the inputs.

<sup>2</sup> Except for Section 5.7.5, we recap experiments designed and conducted by Lukas Schneider (Cordts et al., 2017b; L. Schneider et al., 2016). Note that the FCNs providing the pixel-level semantic labeling input channels are based on Section 4.3.5 and are a contribution of this dissertation.

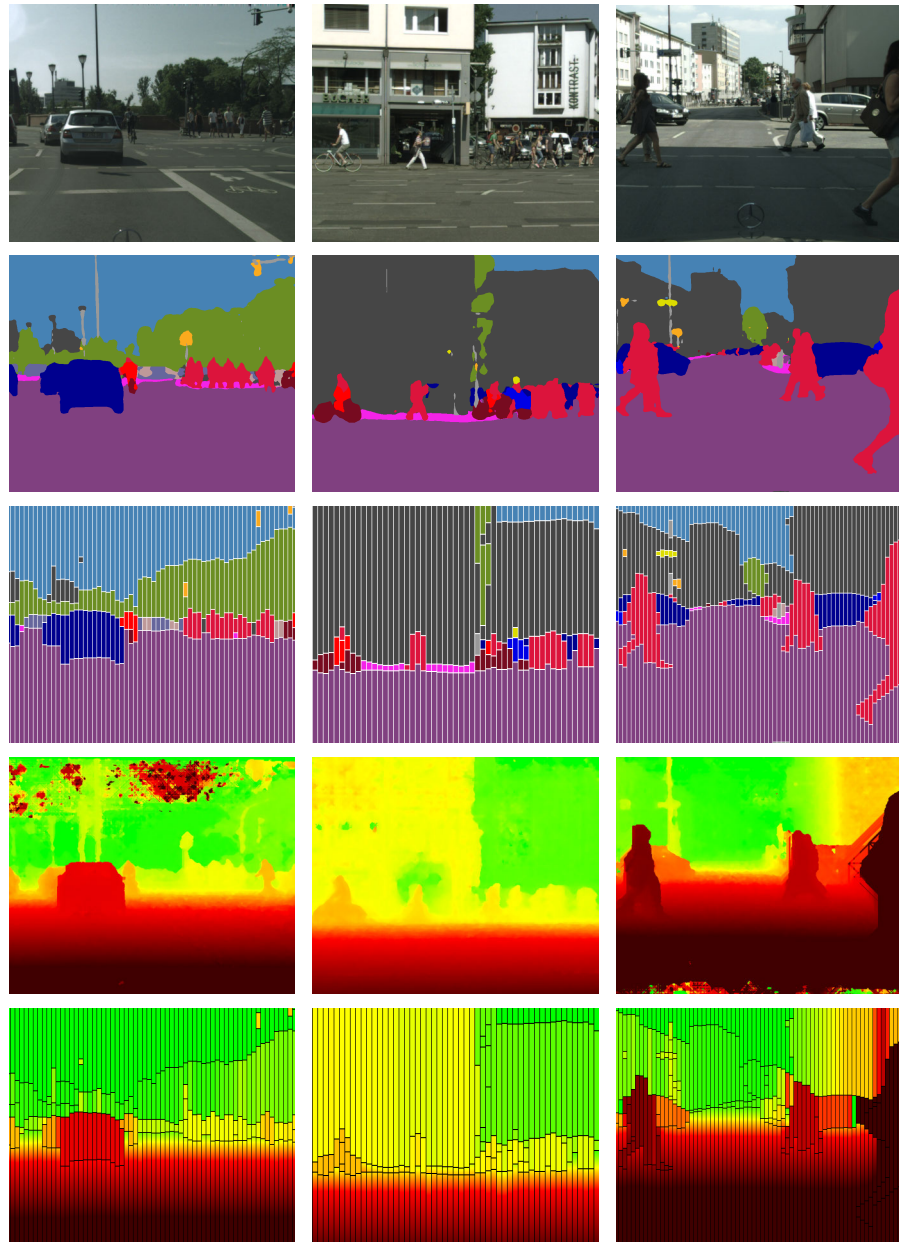


Figure 5.7: Exemplary results of our Semantic Stixel model on Cityscapes (Chapter 4). We show the semantic and depth representation using the same color scheme as in Figure 5.1. From top to bottom: input image, semantic input channel, semantic Stixel representation, depth input channel, depth Stixel representation. Note that the segmentation even captures small objects such as traffic signs and lights. Courtesy of Lukas Schneider.

### 5.7.1 Datasets

We rely on the subset of [KITTI](#) (Geiger et al., 2013) annotated by Ladický et al., 2014 as our first evaluation dataset and denoted as *Ladicky* in the following. This dataset contains both, semantic labeling and depth ground truth, and thus allows for a joint evaluation of both output aspects. We use all 60 images for evaluation and none for training. As suggested by Ladický et al., 2014, 8 annotated classes are used for evaluation and the three rarest classes are ignored.

For training the [FCN](#) that we use during experiments on the *Ladicky* dataset, we leverage other parts of [KITTI](#) that were annotated by various research groups (H. He and Upcroft, 2013; Kundu et al., 2014; Ros et al., 2015; Sengupta et al., 2013; P. Xu et al., 2013; R. Zhang et al., 2015). Their annotations are mapped onto the 8 classes in *Ladicky*, yielding a training set consisting of 676 images. Motivated by the observations in Section 4.3.6, we initialize the training with a model trained on Cityscapes. In doing so, the classification performance of the stand-alone [FCN](#) input channel is improved by 6 % [IoU](#) compared to an [FCN](#) without Cityscapes initialization.

In addition to *Ladicky*, we extend our evaluation to two datasets that only allow for an analysis of depth and classification performance individually, since there is no ground truth for the respective other modality available. First, we use the training set belonging to the stereo challenge in [KITTI'15](#) (Menze and Geiger, 2015) consisting of 200 images to report disparity accuracy. Second, we evaluate pixel-level semantic labeling performance on the validation set of Cityscapes, c.f. Chapter 4.

### 5.7.2 Metrics

For quantitative analysis, we use four performance metrics that analyze distinct aspects of the Semantic Stixel model as well as of our baseline method. First, depth accuracy is reported as the percentage of inlier disparity estimates according to the definition by Menze and Geiger, 2015. An estimated disparity is considered as an inlier, if the absolute deviation compared with ground truth is less than or equal to 3 px and simultaneously the relative deviation is smaller than 5 %. Second, semantic labeling accuracy is measured via the Intersection-over-Union ([IoU](#)) metric averaged over all classes and defined in Section 3.8.3. Third, we determine the run time of the approach, since real-time capability is crucial for autonomous driving. As compute hardware, we employ a 3 GHz, 10 core, Intel Xeon [CPU](#), an NVIDIA Titan X (Maxwell) [GPU](#) for the [FCN](#), as well as an [FPGA](#) for [SGM](#). Last, we aim to measure the compression capabilities of the model and use the number of segments per image as a proxy metric for the representation complexity.

### 5.7.3 Baseline

To allow for an intuitive understanding of quantitative performance numbers, we include a simple baseline model in our analysis that we refer to as *smart downsampling*. As the name suggests, this reference model produces a downsampled version of the disparity and semantic labeling image. The downsampling factor is selected such that the Stixel segmentation and the baseline have the same complexity, measured by the number of bytes required to encode the individual representations. For example, on [KITTI](#) the downsampling factor is chosen as 21, which matches the complexity of 700 Stixels as obtained on average when computed on this dataset.

Downsampling of the semantic labeling image is conducted by average pooling of the classification scores. However, applying a simple pooling to the disparity image would lead to bad results in ground regions. Hence, we make this process *smart* and apply the ground model of the Stixel segmentation. We use the semantic labels to determine ground regions, subtract the ground model from the disparities, and then compute the average. For sky pixels, we assign disparity values of zero.

### 5.7.4 Impact of input cues

Our Semantic Stixel model is based on three input modalities, i.e. depth maps  $\mathbf{D}$ , the color image  $\mathbf{I}$ , and semantic label scores  $\mathbf{L}$ . In this section, we aim to assess their individual contributions in ablation studies. Therefore, we evaluate the Stixel representations computed using only a single input source at a time, as well as different combinations. As we always seek for a representation of depth and semantics, we assign this data in a post-processing step if the respective cue is not used during Stixel inference. Qualitative results of these studies are shown in Figure 5.8 and a quantitative analysis is provided in Table 5.1. Overall, we observe that semantic and color information helps to increase the depth accuracy, while depth and color channels improve semantic labeling performance. Interestingly, the gain obtained by exploiting the color image is fairly small, in particular when semantic input is included. We account this to the powerful [FCN](#) that already captures important image edges and hence adding the latter helps only marginally. Note further that an improved performance in terms of depth and semantic accuracy comes with a slightly increased number of Stixels (last row in Table 5.1). Surprisingly, the smart downsampling baseline performs reasonably well on [Ladicky](#) and [KITTI'15](#) at an output image size of only  $58 \times 18$  pixels. However, on [Cityscapes](#), with more complex scenes and more precise ground truth, the performance is worse. Nevertheless, the quantitative metrics can only capture parts of the draw-backs and the true differences

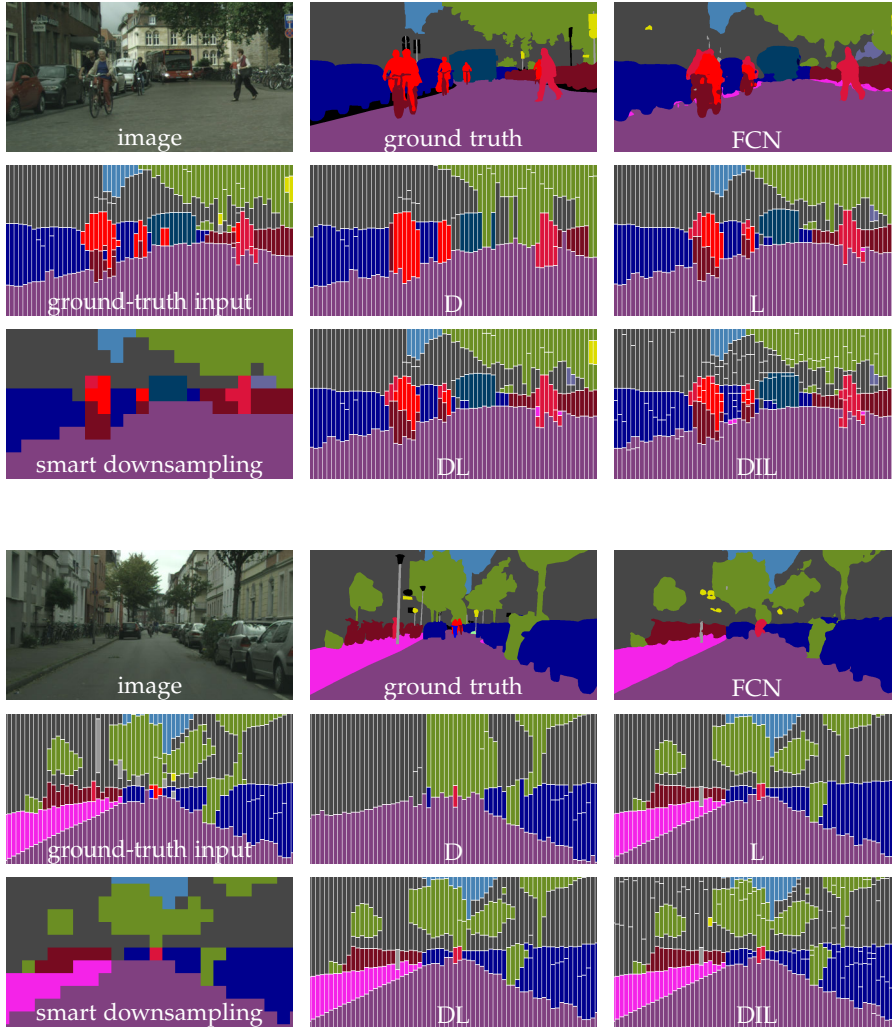


Figure 5.8: Qualitative results of several ablation studies. We show the obtained semantic labeling of the Semantic Stixel segmentation computed using different input channels (**D**epth, **C**olor **I**mage, **S**emantic **L**abels). In addition, we show ground truth, the **FCN** input channel, and the *smart downsampling* baseline. Without the depth channel (**L**), objects of the same semantic class cannot be separated, c.f. the cars in the front. When removing semantic cues (**D**), regions are not accurately segmented, e.g. the distant bus (top) or the sidewalk (bottom). The color image helps in particular when an over-segmentation can be tolerated but a precise segmentation of small objects is desirable, e.g. the traffic light (bottom). The smart downsampling baseline fails to represent small objects which is more pronounced in this qualitative study rather than in the quantitative results in Table 5.1. Courtesy of Lukas Schneider.



Metric	Data	SGM FCN	SDS	D	I	L	DI	DL	IL	DIL
Disp. acc. [%]	LA	82.4	82.7	80.0	47.9	72.7	80.9	83.0	72.6	<b>83.3</b>
	KI	90.6	88.9	90.4	58.7	77.5	90.7	91.3	81.9	<b>91.4</b>
IoU [%]	LA	<b>69.8</b>	65.8	46.1	27.2	62.5	47.3	66.1	66.8	66.5
	CS	<b>60.8</b>	54.1	43.8	18.1	59.7	44.2	60.0	59.8	60.1
Run time [ms]	LA	39.2	—	24.2	2.7	25.5	24.7	45.7	25.7	46.6
	CS	110	—	70.5	10.5	86.1	70	143	86.4	148
No. of Stixels	KI	0.5 M <sup>a</sup>	745 <sup>b</sup>	509	379	453	652	625	577	745
	CS	2 M <sup>a</sup>	1395 <sup>b</sup>	1131	1254	1048	1444	1382	1283	1395

<sup>a</sup> We list the number of pixels for **SGM** and **FCN** raw data to approximately compare the complexity to Stixels.

<sup>b</sup> We use a downsampling factor that results in the same number of bytes to encode this representation as for the given number of Stixels.

Table 5.1: Quantitative performance analysis of Semantic Stixel model based on varying combinations of the input channels. We report numbers for all combinations of Depth, Color Image, and Semantic Labels and compare to the raw input channels, i.e. **SGM** and **FCN**, as well as the *smart downsampling* baseline (SDS). Performance is reported for three datasets (Section 5.7.1), i.e. Ladicky (LA), **KITTI**'15 (KI), and Cityscapes (CS) based on the four metrics introduced in Section 5.7.2. Run time is the total time to parse a single image including the computation of the input channels. To achieve a higher throughput at the cost of one frame delay, we can pipeline the components. **SGM** and **FCN** run in parallel to the Stixel algorithm, on **FPGA**, **GPU**, and **CPU**, respectively. In doing so, we achieve a frame rate of 47.6 Hz on **KITTI** and 15.4 Hz on Cityscapes for the DIL variant.

in representational capabilities become evident when inspecting the qualitative results in Figure 5.8.

Since disparities and pixel labels are the driving input channels, we now assess the effects when varying their relative contribution to the Stixel segmentation. To this end, we sweep the semantic labeling weight  $\delta_L$ , c.f. Equation (5.19), while keeping the disparity influence  $\delta_D$ , c.f. Equation (5.15), constant at 1. We run the experiment in two different setups, first computing Stixels of width 2 and then of our default width 8. Narrower Stixels yield higher depth and semantic accuracy, but are also slower to compute. The results in Figure 5.9 show that an increased semantic influence naturally improves semantic labeling performance, but surprisingly to some extent also helps to increase the disparity accuracy. We find a weight of 5 to produce the best trade-off between both metrics. In case of the narrow Stixel variant, a high influence of the semantic channel even harms the labeling performance, indicating benefits of depth cues for this modality. We account these effects to the Stixel model acting as a regularizer that can compensate for some errors in the input channels. Furthermore,



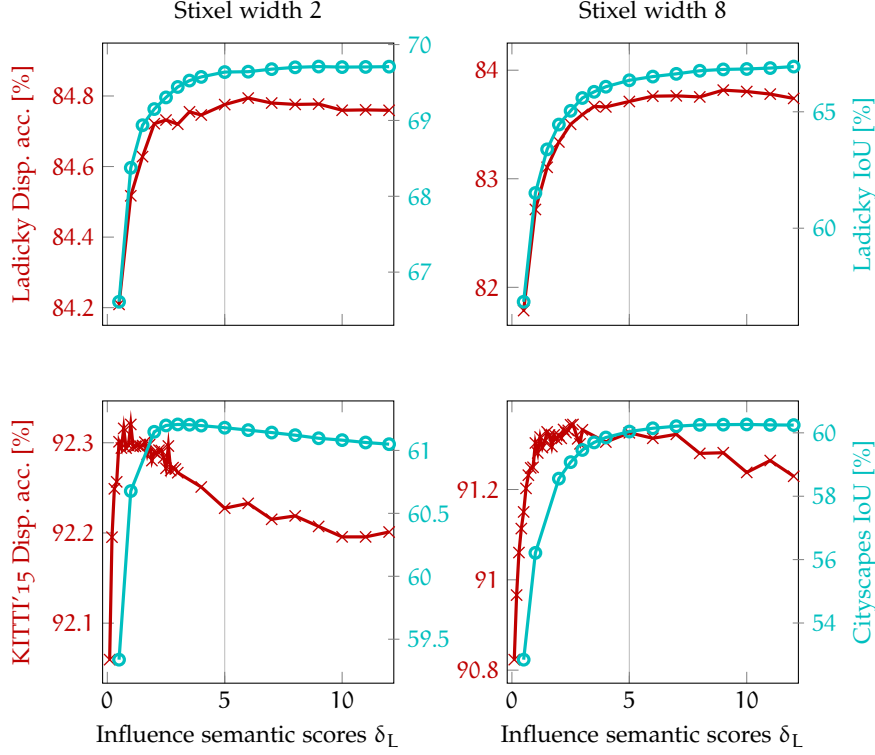


Figure 5.9: Analysis of the influence of the semantic input channel on the disparity and semantic labeling accuracies. We vary the semantic weight  $\delta_L$  while keeping the disparity weight constant at a value of 1. In the left column, we show results for Stixels of width 2, i.e. high accuracy but slow two compute. In the right, we show the performance when using our default width 8. We report two metrics on three datasets, c.f. Sections 5.7.1 and 5.7.2: (1) Disparity accuracy on Ladicky (red, top), (2) IoU on Ladicky (blue, top), (3) Disparity accuracy on KITTI'15 (red, bottom), and (4) IoU on Cityscapes (blue, bottom). Courtesy of Lukas Schneider.

these experiments confirm our findings in Chapter 3 that multiple complementary input cues aid scene recognition performance.

#### 5.7.5 Impact of approximations

In Section 5.4.3, we introduced approximations during inference to keep asymptotic and practical run time low. In our final experiment, we compare these approximations with an exact inference of our model on Ladicky, where the number and size of the images is small enough to conduct this experiment. As is evident from Table 5.2, both variants yield the same result quality, supporting the validity of our approximations. However, the run time of the approximate Stixel inference is two orders of magnitude lower.

	approx. inference	exact inference
Disparity accuracy [%]	83.3	83.1
IoU [%]	66.5	66.3
No. of Stixels per image	745	741
Inference run time [ms]	7.4	863

Table 5.2: Performance of our approximated inference, c.f. Section 5.4.3, compared to an exact inference of Equation (5.20). While the accuracy of both variants is identical, the run time of our inference scheme is two orders of magnitude lower.

## 5.8 DISCUSSION

In this chapter, we introduced Semantic Stixels, a medium-level representation of street scenes combining semantic and depth data. The model focuses on the representation of street scenes for autonomous driving in complex urban scenarios. Based on the assumption that road scenes are geometrically mainly structured in the vertical direction, Stixels model the environment as consisting of support (ground), vertical (obstacles), and sky regions. The Stixel model is founded on a graphical model that provides a clear formalism and allows for a clean integration of different input channels. Based on this model, we derive an efficient inference scheme and show a concept to learn the model parameters from training data. Experiments confirm that depth and semantic cues from object detectors or pixel-level semantic labeling complement each other and the Stixel segmentation effectively integrates both modalities. Overall, we obtain an accurate urban scene representation that is compact and efficient to compute.

## DISCUSSION AND OUTLOOK

## CONTENTS

---

6.1 Discussion . . . . .	137
6.2 Future Work . . . . .	139

---

In this dissertation, we worked towards a system for semantic scene understanding for autonomous driving. In Chapter 3, we learned that multiple complementary input cues significantly aid pixel-level semantic labeling and we found that best performance is achieved when being based on Stixel superpixels. Next, in Chapter 4, we investigated semantic scene understanding when leveraging large-scale data and benchmarked state-of-the-art methods. Based on our findings, we proposed an efficient yet competitive FCN for pixel-level semantic labeling. Subsequently, we combined the lessons learned in Chapter 5. We proposed Semantic Stixels that integrate our real-time semantic labeling FCN with depth information from stereo vision and yield a compact medium-level representation of road scenes.

We now explicitly address the research questions that we initially posed in Section 1.4 and summarize the lessons learned in Section 6.1. Next, we conclude the dissertation with an outlook on possible future work in Section 6.2.

## 6.1 DISCUSSION

**POTENTIAL OF MULTIPLE INPUT MODALITIES** When considering autonomous driving, there are multiple input modalities available that could be exploited for autonomous driving. The parsed images typically stem from a video stream providing temporal information. Often, stereo cameras or laser scanners are employed to sense depth information. Also, bounding box object detectors are established components in ADAS that provide additional semantic cues. In order to investigate the benefits of such data, we proposed a system in Chapter 3 that aims to exploit multi-cue information holistically throughout all processing stages. We found that multi-modal data indeed aids semantic labeling performance. Motivated by these findings, we ensured that multi-cue data is also available in Cityscapes, a large-scale dataset and benchmark suite presented in Chapter 4. We included video, stereo, vehicle odometry, and GPS data to facilitate

continued research in that domain. Via the graphical model that defines the Semantic Stixel world in Chapter 5, we enabled a flexible formulation that allows to integrate various input channels. We showed that depth, color and semantic information in the form of bounding box detections and pixel-level labels can be combined to achieve best performance.

**KNOWN SCENE TYPE** In the context of urban scene understanding, the variance of occurring scene types is limited. The images always contain street scenes, recorded in inner-city traffic, while the camera is mounted in a vehicle at constant position with known pose. We leveraged this knowledge throughout the dissertation either via manual modeling or via machine learning. In Chapter 3, we designed features that encode multi-cue information specifically for street scene understanding. Based on known camera pose, we extracted features such as height above ground or velocity that are specifically discriminative for street scenes. When pairing large-scale data with deep learning methods in Chapter 4, we observe that the classifier can separate road from sidewalk regions. As their texture is very similar, we account these capabilities to prior information about the scene type that the network learned during training. The Semantic Stixels in Chapter 5, on the other hand, are specifically designed for the scene type at hand by describing the scene in terms of support, vertical, and sky regions. In addition, the model contains prior terms that model the common scene layout. We also show how the model's parameters can in principle be learned from training data.

**LARGE-SCALE URBAN SCENE UNDERSTANDING** Inspired by the success of large-scale datasets for general image parsing, we investigated their importance for urban scene understanding. To this end, we proposed the Cityscapes dataset and benchmark suite in Chapter 4 that is the largest and most diverse dataset of street scenes with high-quality and coarse annotations to date. We extensively analyzed the dataset and described our major design choices towards the goal of a suitable dataset for autonomous driving. Based on the dataset, significant progress for urban scene understanding was made and the benchmark allows for a transparent and fair evaluation of suitable methods. Keeping the scope of this dissertation in mind, we combined established approaches to obtain an efficient and competitive FCN for pixel-level semantic labeling.

**SCENE REPRESENTATION** A constant trend towards increasing image resolutions creates significant computational challenges for processing in autonomous vehicles. In Chapter 4, we presented the Cityscapes dataset that consists of images with a resolution of 2 MP and we showed that this high resolution in fact increases recognition

performance. In order to make this high-resolution data available to subsequent processing stages, we argue that we need an abstraction and compression of the raw data while keeping the accuracy high. To this end, we introduce Semantic Stixels in Chapter 5 that parse the scene at real-time speeds and provide a representation with the desired properties. Simultaneously, the representation is robust due to the incorporation of prior knowledge and combines a 3D with a semantic scene understanding. We even argue that Stixels mitigate the need for an instance-level scene understanding to some extent. When inspecting Figure 5.1, we observe that the Stixel segments are capable to separate object instances of the same class at the cost of an over-segmentation. In horizontal direction this is enforced by construction, and in the vertical direction depth cues aid the segmentation. Despite the over-segmentation, Stixels provide sufficient spatial support to robustly estimate their motion state, c.f. Section 3.4.1, allowing for behavior prediction without explicit instance-level knowledge.

## 6.2 FUTURE WORK

We now go beyond the scope of this dissertation and discuss potential future work that could be conducted based on our findings. We explicitly address potential shortcomings of this dissertation and raise open questions that should be answered via future research.

**MULTI-CUE DEEP LEARNING** Deep learning is the state-of-the-art machine learning method for classification and many other tasks. However, when it comes to leveraging multiple input modalities for semantic scene understanding, it becomes challenging to achieve performance gains that are proven to be driven by multi-cue data. Possibly this is due to the established training of neural networks via Stochastic Gradient Descent (SGD), where the greedy solution is to focus on the color image and to ignore additional modalities. The latter are more likely to aid generalization, but are less helpful to greedily classify a single datum. Another possible explanation is the lack of large-scale multi-modal data for general image parsing, i.e. an ImageNet equivalent with multi-cue inputs that could be used for model initialization.

With Cityscapes, presented in Chapter 4, we ensured to create a benchmark that offers various input modalities, but so far, there are only very few methods using multi-cue data at all, and their performance is not yet state of the art. In particular, the time domain that seems to be a driving force for human perception as well as High Dynamic-Range (HDR) images with additional sensory information are barely considered. Recently, Neverova et al., 2017 used Cityscapes to learn the behavior of objects in the scene and to predict their future state. In doing so, the neural network has to become aware of

time and we hope that such work motivates other researchers to follow similar paths. Throughout this dissertation, we studied multi-cue models extensively and confirmed that multiple modalities help, but we relied on classic techniques (Chapter 3) or combined them with single-cue deep learning (Chapter 5). In the future, we plan to bridge the gap and to develop methods that fuse the modalities early to maximally exploit the available information.

**EXTENSIONS OF CITYSCAPES** We see the Cityscapes dataset and benchmark suite (Chapter 4) as a dynamic and ongoing project that we will adapt to novel developments and needs over time to keep pace with the rapid advances in urban semantic scene understanding. Recently, S. Zhang et al., 2017 annotated bounding boxes around humans in the Cityscapes dataset and include occlusion as well as activity information. We plan to incorporate these annotations into the dataset’s website and to extend the benchmark with a pedestrian detection task. Other possible future extensions include adding data from more cities, ideally from all over the world (c.f. Y.-H. Chen et al., 2017), or incorporating more diverse weather and illumination scenarios, e.g. rain, night, or snow.

Another planned direction of future research is to re-address the instance-level semantic labeling task. The current challenge is inspired by an object detection perspective, and predictions are represented as a set of detections that have confidence scores and can possibly overlap. The only difference to classic detection challenges is that each prediction is associated with a segmentation instead of a bounding box. In contrast, from a segmentation perspective, an instance-level prediction would be a segmentation of the scene into instances, i.e. each pixel would be associated with exactly one instance or the background. In such a setup, predictions are non-overlapping and confidence scores are rather unnatural as segmentation is a multi-class problem whereas confidences refer to binary decisions. To account for this second perspective and for the fact that many researchers actually addressed instance-level labeling on Cityscapes from a segmentation perspective, we plan to create a second instance-level challenge. We will develop suitable metrics that fairly assess the performance of segmentation-centered approaches, re-evaluate existing submissions using these metrics, and in doing so provide the baselines for future methods. Note that it is also natural to combine such an instance-level task with pixel-level labeling, where traffic participants are encoded as individual instances and background classes via semantic segments.

**SEMANTIC STIXEL MODEL** In Chapter 5, we introduced a graphical model that defines the Stixel segmentation problem. However, it remains a challenging task to determine the optimal parameteri-

zation that is robust to diverse weather scenarios and produces best results for different tasks and downstream applications. Thus, we investigated automated parameter learning via [S-SVMs](#) in [Section 5.5](#), but we could not outperform a manually modeled parameterization yet. Nevertheless, this research should be continued in the future to allow for possible extensions, such as an online parameter learning to adapt to changing environment conditions.

Our Semantic Stixel model effectively combines semantic class information with depth cues. As discussed in [Section 6.1](#), this combination is capable of achieving an instance-aware representation to some extent. However, appearance information delineating individual instances of the same semantic class, e.g. stemming from an instance-level labeling system, is ignored. In the future, we plan to extend the Stixel segmentation based on the graphical model to incorporate such knowledge. Ultimately, we would obtain a representation that encodes the full information of a visual perception system for autonomous driving.





## APPENDIX

## CONTENTS

A.1	Cityscapes Label Definitions . . . . .	143
A.1.1	Human . . . . .	143
A.1.2	Vehicle . . . . .	144
A.1.3	Nature . . . . .	145
A.1.4	Construction . . . . .	145
A.1.5	Object . . . . .	146
A.1.6	Sky . . . . .	147
A.1.7	Flat . . . . .	147
A.1.8	Void . . . . .	148

A.1 CITYSCAPES LABEL DEFINITIONS<sup>1</sup>

This section provides precise definitions of our annotated classes. These definitions were used to guide our labeling process, as well as quality control, c.f. Section 4.2.2. In addition, we include a typical example for each class.

A.1.1 *Human*

**PERSON** All humans that would primarily rely on their legs to move if necessary. Consequently, this label includes people who are standing/sitting, or otherwise stationary. This class also includes babies, people pushing a bicycle, or standing next to it with both legs on the same side of the bicycle.



<sup>1</sup> The creation of the Cityscapes dataset including the label definitions was initially driven by Marius Cordts and was then continued as joint work with Mohamed Omran. The definitions in this section are based on joint work of Mohamed Omran and Marius Cordts in discussions with the other authors of the publication Cordts et al., 2016. This section contains verbatim quotes of the supplemental material in that publication.

**RIDER** Humans relying on some device for movement. This includes drivers, passengers, or riders of bicycles, motorcycles, scooters, skateboards, horses, Segways, (inline) skates, wheelchairs, road cleaning cars, or convertibles. Note that a visible driver of a closed car can only be seen through the window. Since holes are considered part of the surrounding object, the human is included in the *car* label.



#### A.1.2 Vehicle

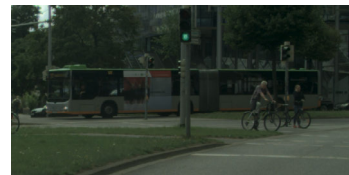
**CAR** This includes cars, jeeps, SUVs, vans with a continuous body shape (i.e. the driver's cabin and cargo compartment are one). Does not include trailers, which have their own separate class.



**TRUCK** This includes trucks, vans with a body that is separate from the driver's cabin, pickup trucks, as well as their trailers.



**BUS** This includes buses that are intended for 9+ persons for public or long-distance transport.



**TRAIN** All vehicles that move on rails, e.g. trams, trains.



**MOTORCYCLE** This includes motorcycles, mopeds, and scooters without the driver or other passengers. The latter receive the label *rider*.



**BICYCLE** This includes bicycles without the cyclist or other passengers. The latter receive the label *rider*.



**CARAVAN** Vehicles that (appear to) contain living quarters. This also includes trailers that are used for living and has priority over the *trailer* class.



**TRAILER** Includes trailers that can be attached to any vehicle, but excludes trailers attached to trucks. The latter are included in the *truck* label.



### A.1.3 Nature

**VEGETATION** Trees, hedges, and all kinds of vertically growing vegetation. Plants attached to buildings/walls/fences are not annotated separately, and receive the same label as the surface they are supported by.



**TERRAIN** Grass, all kinds of horizontally spreading vegetation, soil, or sand. These are areas that are not meant to be driven on. This label may also include a possibly adjacent curb. Single grass stalks or very small patches of grass are not annotated separately and thus are assigned to the label of the region they are growing on.



### A.1.4 Construction

**BUILDING** Includes structures that house/shelter humans, e.g. low-rises, skyscrapers, bus stops, car ports. Translucent buildings made of glass still receive the label *building*. Also includes scaffolding attached to buildings.



**WALL** Individually standing walls that separate two (or more) outdoor areas, and do not provide support for a building.



**FENCE** Structures with holes that separate two (or more) outdoor areas, sometimes temporary.



**GUARD RAIL** Metal structure located on the side of the road to prevent serious accidents. Rare in inner cities, but occur sometimes in curves. Includes the bars holding the rails.



**BRIDGE** Bridges (on which the ego-vehicle is not driving) including everything (fences, guard rails) permanently attached to them.



**TUNNEL** Tunnel walls and the (typically dark) space encased by the tunnel, but excluding vehicles.



#### A.1.5 *Object*

**TRAFFIC SIGN** Front part of signs installed by the state/city authority with the purpose of conveying information to drivers/cyclists/pedestrians, e.g. traffic signs, parking signs, direction signs, or warning reflector posts.



**TRAFFIC LIGHT** The traffic light box without its poles in all orientations and for all types of traffic participants, e.g. regular traffic light, bus traffic light, train traffic light.



**POLE** Small, mainly vertically oriented poles, e.g. sign poles or traffic light poles. This does not include objects mounted on the pole, which have a larger diameter than the pole itself (e.g. most street lights).



**POLE GROUP** Multiple poles that are cumbersome to label individually, but where the background can be seen in their gaps.



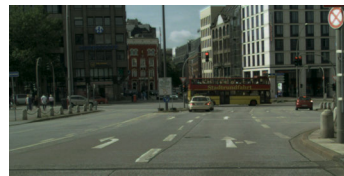
#### A.1.6 *Sky*

**SKY** Open sky (without tree branches/leaves)



#### A.1.7 *Flat*

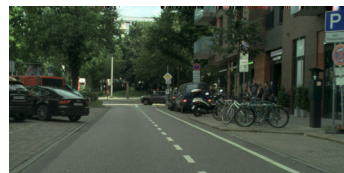
**ROAD** Horizontal surfaces on which cars usually drive, including road markings. Typically delimited by curbs, rail tracks, or parking areas. However, *road* is not delimited by road markings and thus may include bicycle lanes or roundabouts, but not curbs.



**SIDEWALK** Horizontal surfaces designated for pedestrians or cyclists. Delimited from the road by some obstacle, e.g. curbs or poles (might be small), but not only by markings. Often elevated compared to the road and often located at the side of a road. The curbs are included in the *sidewalk* label. Also includes the walkable part of traffic islands, as well as pedestrian-only zones, where cars are not allowed to drive during regular business hours. If it's an all-day mixed pedestrian/car area, the correct label is *ground*.



**PARKING** Horizontal surfaces that are intended for parking and separated from the road, either via elevation or via a different texture/material, but not separated merely by markings.





**RAIL TRACK** Horizontal surfaces on which only rail cars can normally drive. If rail tracks for trams are embedded in a standard road, they are included in the *road* label.



#### A.1.8 *Void*

**GROUND** All other forms of horizontal ground-level structures that do not match any of the above, for example mixed zones (cars and pedestrians), roundabouts that are flat but delimited from the road by a curb, or in general a fallback label for horizontal surfaces that are difficult to classify, e.g. due to having a dual purpose.



**DYNAMIC** Movable objects that do not correspond to any of the other non-void categories and might not be in the same position in the next day/hour/minute, e.g. movable trash bins, buggies, luggage, animals, chairs, or tables.



**STATIC** This includes areas of the image that are difficult to identify/label due to occlusion/distance, as well as non-movable objects that do not match any of the non-void categories, e.g. mountains, street lights, reverse sides of traffic signs, or permanently mounted commercial signs.



**EGO VEHICLE** Since a part of the vehicle from which our data was recorded is visible in all frames, it is assigned to this special label. This label is also available at test time.

**UNLABELED** Pixels that were not explicitly assigned to a label.

**OUT OF ROI** Narrow strip of 5 pixels along the image borders that is not considered for training or evaluation. This label is also available at test-time.

**RECTIFICATION BORDER** Areas close to the image border that contain artifacts resulting from the stereo pair rectification. This label is also available at test time.

## BIBLIOGRAPHY

---

- Achanta, Radhakrishna, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk (2012). „SLIC superpixels compared to state-of-the-art superpixel methods.“ In: *Trans. PAMI* 34.11, pp. 2274–2282 (cit. on pp. [xvi](#), [27](#), [110](#), [120](#)).
- Arbeláez, Pablo, Bharath Hariharan, Chunhui Gu, Saurabh Gupta, Lubomir Bourdev, and Jitendra Malik (2012). „Semantic segmentation using regions and parts.“ In: *CVPR*, pp. 3378–3385 (cit. on pp. [14](#), [15](#), [21](#), [111](#)).
- Arbeláez, Pablo, Michael Maire, Charless Fowlkes, and Jitendra Malik (2011). „Contour detection and hierarchical image segmentation.“ In: *Trans. PAMI* 33.5, pp. 898–916 (cit. on pp. [xvi](#), [14](#), [21](#), [22](#), [46](#), [111](#)).
- Ardeshir, Shervin, Kofi Malcolm Collins-Sibley, and Mubarak Shah (2015). „Geo-semantic segmentation.“ In: *CVPR*, pp. 2792–2799 (cit. on pp. [18](#), [68](#)).
- Arnab, Anurag, Sadeep Jayasumana, Shuai Zheng, and Philip H S Torr (2016). „Higher order conditional random fields in deep neural networks.“ In: *ECCV*, pp. 524–540 (cit. on p. [16](#)).
- Badino, Hernán, Uwe Franke, and David Pfeiffer (2009). „The Stixel world - A compact medium level representation of the 3D world.“ In: *DAGM Symposium*, pp. 51–60 (cit. on p. [108](#)).
- Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2017). „SegNet: A deep convolutional encoder-decoder architecture for image segmentation.“ In: *Trans. PAMI* (cit. on pp. [16](#), [57](#), [98](#), [99](#)).
- Benenson, Rodrigo, Markus Mathias, Radu Timofte, and Luc Van Gool (2012). „Fast stixel computation for fast pedestrian detection.“ In: *CVVT Workshop (ECCV)*, pp. 11–20 (cit. on pp. [108](#), [110](#)).
- Benz, Karl Friedrich (1886). *Patent No. 37435* (cit. on p. [6](#)).
- Blake, Andrew and Pushmeet Kohli (2011). „Introduction to Markov Random Fields.“ In: *Markov Random Fields for vision and image processing*. Ed. by Andrew Blake, Pushmeet Kohli, and Carsten Rother. MIT Press, pp. 1–28 (cit. on pp. [15](#), [22](#), [39](#), [40](#)).
- Boureau, Y-Lan, Jean Ponce, and Yann LeCun (2010). „A theoretical analysis of feature pooling in visual recognition.“ In: *ICML*, pp. 111–118 (cit. on p. [14](#)).
- Brostow, Gabriel J, Julien Fauqueur, and Roberto Cipolla (2009). „Semantic object classes in video: A high-definition ground truth database.“ In: *Pattern Recognition Letters* 30.2, pp. 88–97 (cit. on pp. [xiv](#), [58](#), [61](#), [66](#), [68](#), [98](#), [99](#)).

- Brox, Thomas, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik (2011). „Object segmentation by alignment of poselet activations to image contours.“ In: *CVPR*, pp. 2225–2232 (cit. on p. 14).
- Byeon, Wonmin, Thomas M Breuel, Federico Raue, and Marcus Liwicki (2015). „Scene labeling with LSTM recurrent neural networks.“ In: *CVPR*, pp. 3547–3555 (cit. on p. 16).
- Carreira, João, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu (2012a). „Semantic segmentation with second-order pooling.“ In: *ECCV*, pp. 430–443 (cit. on p. 14).
- Carreira, João, Fuxin Li, and Cristian Sminchisescu (2012b). „Object recognition by sequential figure-ground ranking.“ In: *IJCV* 98.3, pp. 243–262 (cit. on p. 111).
- Carreira, João and Cristian Sminchisescu (2012). „CPMC: Automatic object segmentation using constrained parametric min-cuts.“ In: *Trans. PAMI* 34.7, pp. 1312–1328 (cit. on p. 111).
- Chen, Yi-Hsin, Wei-Yu Chen, Yu-Ting Chen, Bo-Cheng Tsai, Yu-Chiang Frank Wang, and Min Sun (2017). „No more discrimination: Cross city adaptation of road scene segmenters.“ In: *arXiv:1704.08509v1 [cs.CV]* (cit. on p. 140).
- Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin P Murphy, and Alan L Yuille (2016). „DeepLab: Semantic image segmentation with deep convolutional nets and fully connected CRFs.“ In: *arXiv:1606.00915v1 [cs.CV]* (cit. on pp. 16, 75, 84, 85, 91).
- Chen, Liang-Chieh, George Papandreou, and Alan L Yuille (2013). „Learning a dictionary of shape epitomes with applications to image labeling.“ In: *ICCV*, pp. 337–344 (cit. on p. 14).
- Chen, Xiaozhi, Kaustav Kundu, Yukun Zhu, Andrew Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun (2015). „3D object proposals for accurate object class detection.“ In: *NIPS*, pp. 424–432 (cit. on p. 18).
- Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2016). „Fast and accurate deep network learning by Exponential Linear Units (ELUs).“ In: *ICLR* (cit. on p. 17).
- Comaniciu, Dorin and Peter Meer (2002). „Mean shift: A robust approach toward feature space analysis.“ In: *Trans. PAMI* 24.5, pp. 603–619 (cit. on p. 110).
- Cordts, Marius, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele (2016). „The Cityscapes dataset for semantic urban scene understanding.“ In: *CVPR*, pp. 3213–3223 (cit. on pp. 57, 58, 91, 92, 99, 105, 143).
- Cordts, Marius, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele (2015). „The Cityscapes dataset.“ In:



- CVPR Workshop on *The Future of Datasets in Vision* (cit. on pp. 57, 102).
- Cordts, Marius, Timo Rehfeld, Markus Enzweiler, Uwe Franke, and Stefan Roth (2017a). „Tree-structured models for efficient multi-cue scene labeling.“ In: *Trans. PAMI* 39.7, pp. 1444–1454 (cit. on pp. 19, 22, 25, 26).
- Cordts, Marius, Timo Rehfeld, Lukas Schneider, David Pfeiffer, Markus Enzweiler, Stefan Roth, Marc Pollefeys, and Uwe Franke (2017b). „The Stixel world: A medium-level representation of traffic scenes.“ In: *IVC* (cit. on pp. 109, 120, 124, 129).
- Cordts, Marius, Lukas Schneider, Markus Enzweiler, Uwe Franke, and Stefan Roth (2014). „Object-level priors for Stixel generation.“ In: *GCPR*, pp. 172–183 (cit. on pp. 110, 126, 128, 129).
- Couprrie, Camille, Clément Farabet, Laurent Najman, and Yann LeCun (2013). „Indoor semantic segmentation using depth information.“ In: *ICLR* (cit. on p. 18).
- Daniel, Malcolm (2004). *Daguerre (1787–1851) and the invention of photography*. URL: [http://www.metmuseum.org/toah/hd/dagu/hd%7B%5C\\_%7Ddagu.htm](http://www.metmuseum.org/toah/hd/dagu/hd%7B%5C_%7Ddagu.htm) (visited on 12/20/2016) (cit. on p. 1).
- Deng, Zhuo, Sinisa Todorovic, and Longin Jan Latecki (2015). „Semantic segmentation of RGBD images with mutex constraints.“ In: *ICCV*, pp. 1733–1741 (cit. on p. 18).
- Dhiman, Vikas, Abhijit Kundu, Frank Dellaert, and Jason J Corso (2014). „Modern MAP inference methods for accurate and fast occupancy grid mapping on higher order factor graphs.“ In: *ICRA*, pp. 2037–2044 (cit. on p. 111).
- Dollár, Piotr, Christian Wojek, Bernt Schiele, and Pietro Perona (2012). „Pedestrian detection: An evaluation of the state of the art.“ In: *Trans. PAMI* 34.4, pp. 743–761 (cit. on pp. 68, 71).
- Eigen, David and Rob Fergus (2015). „Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture.“ In: *ICCV*, pp. 2650–2658 (cit. on p. 18).
- Eitel, Andreas, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard (2015). „Multimodal deep learning for robust RGB-D object recognition.“ In: *IROS*, pp. 681–687 (cit. on p. 18).
- Enzweiler, Markus and Darius M Gavrila (2009). „Monocular pedestrian detection: Survey and experiments.“ In: *Trans. PAMI* 31.12, pp. 2179–2195 (cit. on pp. 41, 49, 125).
- Enzweiler, Markus, Matthias Hummel, David Pfeiffer, and Uwe Franke (2012). „Efficient Stixel-based object recognition.“ In: *IV Symposium*, pp. 1066–1071 (cit. on pp. 18, 108).
- Erbs, Friedrich, Beate Schwarz, and Uwe Franke (2012). „Stixmentation: Probabilistic Stixel based traffic scene labeling.“ In: *BMVC*, pp. 71.1–71.12 (cit. on p. 108).

- Everingham, Mark, S M Ali Eslami, Luc Van Gool, Christopher K I Williams, John Winn, and Andrew Zisserman (2014). „The PASCAL visual object classes challenge: A retrospective.“ In: *IJCV* 111.1, pp. 98–136 (cit. on pp. [xvi](#), [3](#), [6](#), [20](#), [39](#), [42](#), [44](#), [57](#), [59](#), [68](#), [71](#), [73](#), [101](#)).
- Farabet, Clément, Camille Couprie, Laurent Najman, and Yann LeCun (2012). „Learning hierarchical features for scene labeling.“ In: *Trans. PAMI* 35.8, pp. 1915–1929 (cit. on pp. [16](#), [21](#), [98](#)).
- Fawcett, Tom (2006). „An introduction to ROC analysis.“ In: *Pattern Recognition Letters* 27.8, pp. 867–871 (cit. on p. [43](#)).
- Felzenszwalb, Pedro F, Ross B Girshick, David McAllester, and Deva Ramanan (2010). „Object detection with discriminatively trained part based models.“ In: *Trans. PAMI* 32.9, pp. 1627–45 (cit. on p. [125](#)).
- Felzenszwalb, Pedro F and Daniel P Huttenlocher (2004). „Efficient graph-based image segmentation.“ In: *IJCV* 59.2, pp. 167–181 (cit. on pp. [xv](#), [14](#), [21](#), [27](#), [29](#), [110](#)).
- Felzenszwalb, Pedro F and Olga Veksler (2010). „Tiered scene labeling with dynamic programming.“ In: *CVPR*, pp. 3097–3104 (cit. on pp. [17](#), [112](#)).
- Flickr (2016). *Website*. URL: <http://www.flickr.com/> (cit. on p. [20](#)).
- Floros, Georgios and Bastian Leibe (2012). „Joint 2D-3D temporally consistent semantic segmentation of street scenes.“ In: *CVPR*, pp. 2823–2830 (cit. on pp. [18](#), [21](#)).
- Franke, Uwe, David Pfeiffer, Clemens Rabe, Carsten Knöppel, Markus Enzweiler, Fridtjof Stein, and Ralf G Herrtwich (2013). „Making Bertha see.“ In: *ICCV Workshops*, pp. 214–221 (cit. on pp. [17](#), [21](#), [52](#), [57](#), [108](#), [125](#)).
- Franke, Uwe, Clemens Rabe, Hernán Badino, and Stefan K Gehrig (2005). „6D-Vision: Fusion of stereo and motion for robust environment perception.“ In: *DAGM Symposium*, pp. 216–223 (cit. on p. [41](#)).
- Freund, Yoav and Robert E Schapire (1996). „Experiments with a new boosting algorithm.“ In: *ICML*, pp. 148–156 (cit. on p. [15](#)).
- Fröhlich, Björn, Erik Rodner, and Joachim Denzler (2012). „Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach.“ In: *ACCV*, pp. 218–231 (cit. on pp. [14](#), [24](#)).
- Fulkerson, Brian, Andrea Vedaldi, and Stefano Soatto (2009). „Class segmentation and object localization with superpixel neighborhoods.“ In: *ICCV*, pp. 670–677 (cit. on pp. [14](#), [15](#)).
- Furgale, Paul et al. (2013). „Toward automated driving in cities using close-to-market sensors: An overview of the V-Charge project.“ In: *IV Symposium*, pp. 809–816 (cit. on p. [57](#)).

- Gehrig, Stefan K, Felix Eberli, and Thomas Meyer (2009). „A real-time low-power stereo vision engine using semi-global matching.“ In: *ICVS*, pp. 134–143 (cit. on pp. 61, 124, 129).
- Gehrig, Stefan K, Reto Stalder, and Nicolai Schneider (2015). „A flexible high-resolution real-time low-power stereo vision engine.“ In: *ICVS*, pp. 69–79 (cit. on p. 41).
- Geiger, Andreas, Martin Lauer, Christian Wojek, Christoph Stiller, and Raquel Urtasun (2014). „3D traffic scene understanding from movable platforms.“ In: *Trans. PAMI* 36.5, pp. 1012–1025 (cit. on pp. 18, 57).
- Geiger, Andreas, Philip Lenz, Christoph Stiller, and Raquel Urtasun (2013). „Vision meets robotics: The KITTI dataset.“ In: *IJRR* 32.11, pp. 1231–1237 (cit. on pp. xv, 4, 17, 18, 20, 21, 42, 58, 61, 66, 68, 101, 110, 131).
- Geurts, Pierre, Damien Ernst, and Louis Wehenkel (2006). „Extremely randomized trees.“ In: *Machine Learning* 63.1, pp. 3–42 (cit. on p. 26).
- Ghiasi, Golnaz and Charless Fowlkes (2016). „Laplacian pyramid reconstruction and refinement for semantic segmentation.“ In: *ECCV*, pp. 519–534 (cit. on pp. 16, 85, 92).
- Girshick, Ross B (2015). „Fast R-CNN.“ In: *ICCV*, pp. 1440–1448 (cit. on pp. 57, 125).
- Girshick, Ross B, Jeff Donahue, Trevor Darrell, and Jitendra Malik (2014). „Rich feature hierarchies for accurate object detection and semantic segmentation.“ In: *CVPR*, pp. 580–587 (cit. on pp. xvi, 111).
- Gonfaus, Josep M, Xavier Boix, Joost Van de Weijer, Andrew D Bagdanov, Joan Serrat, and Jordi González (2010). „Harmony potentials for joint classification and segmentation.“ In: *CVPR*, pp. 3280–3287 (cit. on p. 15).
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press (cit. on p. 15).
- Gould, Stephen (2012). „DARWIN: A framework for machine learning and computer vision research and development.“ In: *JMLR* 13, pp. 3533–3537 (cit. on pp. 45–50).
- Güney, Fatma and Andreas Geiger (2015). „Displets: Resolving stereo ambiguities using object knowledge.“ In: *CVPR*, pp. 4165–4175 (cit. on pp. 68, 69).
- Gupta, Abhinav, Alexei A Efros, and Martial Hebert (2010). „Blocks world revisited: Image understanding using qualitative geometry and mechanics.“ In: *ECCV*, pp. 482–496 (cit. on p. 17).
- Gupta, Saurabh, Pablo Arbeláez, Ross B Girshick, and Jitendra Malik (2015). „Indoor scene understanding with RGB-D images: Bottom-up segmentation, object detection and semantic segmentation.“ In: *IJCV* 112.2, pp. 133–149 (cit. on p. 18).

- Gupta, Saurabh, Ross B Girshick, Pablo Arbeláez, and Jitendra Malik (2014). „Learning rich features from RGB-D images for object detection and segmentation.“ In: *ECCV*, pp. 345–360 (cit. on pp. 18, 20–22, 46).
- Gupta, Saurabh, Judy Hoffman, and Jitendra Malik (2016). „Cross modal distillation for supervision transfer.“ In: *CVPR*, pp. 2827–2836 (cit. on p. 18).
- Hariharan, Bharath, Pablo Arbeláez, Ross B Girshick, and Jitendra Malik (2014). „Simultaneous detection and segmentation.“ In: *ECCV*, pp. 297–312 (cit. on pp. 45, 99).
- Hazirbas, Caner, Lingni Ma, Csaba Domokos, and Daniel Cremers (2016). „FuseNet: Incorporating depth into semantic segmentation via fusion-based CNN architecture.“ In: *ACCV*, pp. 213–228 (cit. on p. 18).
- He, Hu and Ben Upcroft (2013). „Nonparametric semantic segmentation for 3D street scenes.“ In: *IROS*, pp. 3697–3703 (cit. on pp. 18, 42, 68, 69, 131).
- He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross B Girshick (2017). „Mask R-CNN.“ In: *arXiv:1703.06870v2 [cs.CV]* (cit. on p. 105).
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). „Deep residual learning for image recognition.“ In: *CVPR*, pp. 770–778 (cit. on pp. 84, 85, 95–97).
- Heilbron, Fabian Caba, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles (2015). „ActivityNet: A large-scale video benchmark for human activity understanding.“ In: *CVPR*, pp. 961–970 (cit. on p. 4).
- Hill, Simon (2013). *From J-Phone to Lumia 1020: A complete history of the camera phone*. URL: <http://www.digitaltrends.com/mobile/camera-phone-history/> (visited on 12/20/2016) (cit. on p. 1).
- Hirschmüller, Heiko (2008). „Stereo processing by semiglobal matching and mutual information.“ In: *Trans. PAMI* 30.2, pp. 328–341 (cit. on pp. 41, 61, 124, 129).
- Hoiem, Derek, Alexei A Efros, and Martial Hebert (2007). „Recovering surface layout from an image.“ In: *IJCV* 75.1, pp. 151–172 (cit. on pp. 34, 111, 112).
- Hou, Saihui, Zilei Wang, and Feng Wu (2016). „Deeply exploit depth information for object detection.“ In: *CVPR*, pp. 19–27 (cit. on p. 18).
- Huang, Qixing, Mei Han, Bo Wu, and Sergey Ioffe (2011). „A hierarchical conditional random field model for labeling and segmenting images of street scenes.“ In: *CVPR*, pp. 1953–1960 (cit. on p. 34).
- Iandola, Forrest N, Matthew W Moskewicz, Khalid Ashraf, Song Han, William J Dally, and Kurt Keutzer (2016). „SqueezeNet: AlexNet-

- level accuracy with 50x fewer parameters and <1MB model size.” In: *arXiv:1602.07360v4 [cs.CV]* (cit. on pp. 17, 85, 96).
- Jarboe, Greg (2015). *VidCon 2015 haul: Trends, strategic insights, critical data, and tactical advice*. URL: <http://tubularinsights.com/vidcon-2015-strategic-insights-tactical-advice/> (visited on 12/20/2016) (cit. on p. 2).
- Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B Girshick, Sergio Guadarrama, and Trevor Darrell (2014). „Caffe: Convolutional architecture for fast feature embedding.” In: *ACMMM*, pp. 675–678 (cit. on pp. 80, 96).
- Kähler, Olaf and Ian Reid (2013). „Efficient 3D scene labeling using fields of trees.” In: *ICCV*, pp. 3064–3071 (cit. on p. 21).
- Karpathy, Andrej, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei (2014). „Large-scale video classification with convolutional neural networks.” In: *CVPR*, pp. 1725–1732 (cit. on p. 4).
- Khan, Salman H, Mohammed Bennamoun, Ferdous Sohel, Roberto Togneri, and Imran Naseem (2016). „Integrating geometrical context for semantic labeling of indoor scenes using RGBD images.” In: *IJCV* 117.1, pp. 1–20 (cit. on p. 18).
- Kohli, Pushmeet, L’ubor Ladický, and Philip H S Torr (2009). „Robust higher order potentials for enforcing label consistency.” In: *IJCV* 82.3, pp. 302–324 (cit. on pp. 15, 40).
- Kreso, Ivan, Denis Causevic, Josip Krapac, and Sinisa Segvic (2016). „Convolutional scale invariance for semantic segmentation.” In: *GCPR*, pp. 64–75 (cit. on pp. 18, 85, 91).
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). „ImageNet classification with deep convolutional neural networks.” In: *NIPS*, pp. 1097–1105 (cit. on pp. 57, 96, 97).
- Krüger, Lars, Chistian Wöhler, Alexander Würz-Wessel, and Fridtjof Stein (2004). „In-factory calibration of multiocular camera systems.” In: *SPIE* 5457, *Optical Metrology in Production Engineering*, pp. 126–137 (cit. on p. 59).
- Kundu, Abhijit, Yin Li, Frank Dellaert, Fuxin Li, and James M Rehg (2014). „Joint semantic segmentation and 3D reconstruction from monocular video.” In: *ECCV*, pp. 703–718 (cit. on pp. 18, 68, 69, 131).
- Ladický, L’ubor, Chris Russell, Pushmeet Kohli, and Philip H S Torr (2013). „Associative hierarchical random fields.” In: *Trans. PAMI* 36.6, pp. 1056–1077 (cit. on pp. 14, 15, 21).
- Ladický, L’ubor, Jianbo Shi, and Marc Pollefeys (2014). „Pulling things out of perspective.” In: *CVPR*, pp. 89–96 (cit. on pp. 18, 42, 68, 69, 131).
- Ladický, L’ubor, Paul Sturgess, Chris Russell, Sunando Sengupta, Yalin Bastanlar, William Clocksin, and Philip H S Torr (2010).

- „Joint optimisation for object class segmentation and dense stereo reconstruction.“ In: *BMVC*, pp. 104.1–104.11 (cit. on pp. 45–51).
- Larish, John J (1990). *Electronic photography*. Computer graphics–technology and management series. TAB Professional and Reference Books (cit. on p. 1).
- LeCun, Yann, Yoshua Bengio, and Geoffrey E Hinton (2015). „Deep learning.“ In: *Nature* 521.7553, pp. 436–444 (cit. on p. 57).
- Leibe, Bastian, Nico Cornelis, Kurt Cornelis, and Luc Van Gool (2007). „Dynamic 3D scene analysis from a moving vehicle.“ In: *CVPR*, pp. 1–8 (cit. on pp. 18, 58, 68).
- Lempitsky, Victor, Andrea Vedaldi, and Andrew Zisserman (2011). „A pylon model for semantic segmentation.“ In: *NIPS*, pp. 1485–1493 (cit. on pp. 15, 21).
- Leung, Thomas and Jitendra Malik (2001). „Representing and recognizing the visual appearance of materials using three-dimensional textons.“ In: *IJCV* 43.1, pp. 29–44 (cit. on p. 24).
- Levi, Dan, Noa Garnett, and Ethan Fetaya (2015). „StixelNet: A deep convolutional network for obstacle detection and road segmentation.“ In: *BMVC*, pp. 109.1–109.12 (cit. on p. 110).
- Levinshtein, Alex, Adrian Stere, Kirikos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi (2009). „TurboPixels: Fast superpixels using geometric flows.“ In: *Trans. PAMI* 31.12, pp. 2290–2297 (cit. on p. 110).
- Lewis, David D (1998). „Naive (Bayes) at forty: The independence assumption in information retrieval.“ In: *ECML*, pp. 4–15 (cit. on p. 14).
- Li, Xiaofei, Fabian Flohr, Yue Yang, Hui Xiong, Markus Braun, Shuyue Pan, Keqiang Li, and Dariu M Gavrilă (2016). „A new benchmark for vision-based cyclist detection.“ In: *IV Symposium*, pp. 1028–1033 (cit. on p. 108).
- Li, Xiaoxiao, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang (2017). „Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade.“ In: *CVPR*, to appear (cit. on pp. 17, 85).
- Li, Zhen, Yukang Gan, Xiaodan Liang, Yizhou Yu, Hui Cheng, and Liang Lin (2016). „LSTM-CF: Unifying context modeling and fusion with LSTMs for RGB-D scene labeling.“ In: *ECCV*, pp. 541–557 (cit. on p. 18).
- Lim, Joseph J, Pablo Arbeláez, Chunhui Gu, and Jitendra Malik (2009). „Context by region ancestry.“ In: *ICCV*, pp. 1978–1985 (cit. on pp. 14, 15, 21).
- Lin, Dahua, Sanja Fidler, and Raquel Urtasun (2013). „Holistic scene understanding for 3D object detection with RGBD cameras.“ In: *ICCV*, pp. 1417–1424 (cit. on p. 18).



- Lin, Guosheng, Anton Milan, Chunhua Shen, and Ian Reid (2017). „RefineNet: Multi-path refinement networks for high-resolution semantic segmentation.“ In: *CVPR*, to appear (cit. on pp. 16, 85).
- Lin, Guosheng, Chunhua Shen, Ian Reid, and Anton van den Hengel (2016). „Efficient piecewise training of deep structured models for semantic segmentation.“ In: *CVPR*, pp. 3194–3203 (cit. on pp. 16, 75, 85, 91).
- Lin, Min, Qiang Chen, and Shuicheng Yan (2014). „Network in network.“ In: *ICLR* (cit. on p. 96).
- Lin, Tsung-Yi, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross B Girshick, James Hays, Pietro Perona, Deva Ramanan, C Lawrence Zitnick, and Piotr Dollár (2014). „Microsoft COCO: Common objects in context.“ In: *ECCV*, pp. 740–755 (cit. on pp. xiv, 3, 4, 6, 20, 57, 59, 68, 71, 100, 101).
- Liu, Ming-Yu, Shuoxin Lin, Srikumar Ramalingam, and Oncel Tuzel (2015). „Layered interpretation of street view images.“ In: *RSS*, pp. 1–9 (cit. on pp. 17, 45, 47, 49, 52, 110, 112).
- Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg (2016). „SSD: Single Shot MultiBox Detector.“ In: *ECCV*, pp. 21–37 (cit. on p. 125).
- Liu, Wei, Andrew Rabinovich, and Alexander C Berg (2015). „ParseNet: Looking wider to see better.“ In: *arXiv:1506.04579v2 [cs.CV]* (cit. on p. 16).
- Liu, Ziwei, Xiaoxiao Li, Ping Luo, Chen Change Loy, and Xiaoou Tang (2015). „Semantic image segmentation via deep parsing network.“ In: *ICCV*, pp. 1377–1385 (cit. on p. 16).
- Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). „Fully convolutional networks for semantic segmentation.“ In: *CVPR*, pp. 3431–3440 (cit. on pp. 75–77, 80, 81).
- Lowe, David G (2004). „Distinctive Image Features from Scale-Invariant Keypoints.“ In: *IJCV* 60.2, pp. 91–110 (cit. on p. 14).
- Luo, Wenjie, Yujia Li, Raquel Urtasun, and Richard Zemel (2016). „Understanding the effective receptive field in deep convolutional neural networks.“ In: *NIPS*, pp. 4898–4906 (cit. on p. 98).
- Maire, Michael, Pablo Arbeláez, Charless Fowlkes, and Jitendra Malik (2008). „Using contours to detect and localize junctions in natural images.“ In: *CVPR*, pp. 1–8 (cit. on p. xv).
- Malik, Jitendra, Serge Belongie, Thomas Leung, and Jianbo Shi (2001). „Contour and texture analysis for image segmentation.“ In: *IJCV* 43.1, pp. 7–27 (cit. on p. 14).
- Meeker, Mary (2016). *Internet trends 2016: Code conference*. URL: <http://www.kpcb.com/internet-trends> (visited on 12/20/2016) (cit. on p. 2).
- Menze, Moritz and Andreas Geiger (2015). „Object scene flow for autonomous vehicles.“ In: *CVPR*, pp. 3061–3070 (cit. on p. 131).



- Mičušík, Branislav and Jana Košecká (2009). „Semantic segmentation of street scenes by superpixel co-occurrence and 3D geometry.“ In: *ICCV Workshops*, pp. 625–632 (cit. on pp. 14, 21).
- Miksik, Ondrej, Daniel Munoz, J Andrew Bagnell, and Martial Hebert (2013). „Efficient temporal consistency for streaming video scene analysis.“ In: *ICRA*, pp. 133–139 (cit. on p. 18).
- Moosmann, Frank, Eric Nowak, and Frederic Jurie (2008). „Randomized clustering forests for image classification.“ In: *Trans. PAMI* 30.9, pp. 1632–1646 (cit. on pp. xv, 14, 23, 25).
- Mostajabi, Mohammadreza, Payman Yadollahpour, and Gregory Shakhnarovich (2015). „Feedforward semantic segmentation with zoom-out features.“ In: *CVPR*, pp. 3379–3385 (cit. on p. 16).
- Mottaghi, Roozbeh, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan L Yuille (2014). „The role of context for object detection and semantic segmentation in the wild.“ In: *CVPR*, pp. 891–898 (cit. on pp. 6, 57, 68, 75).
- Muffert, Maximilian, Nicolai Schneider, and Uwe Franke (2014). „Stix-Fusion: A probabilistic Stixel integration technique.“ In: *CRV*, pp. 16–23 (cit. on pp. 108, 111).
- Müller, Andreas C and Sven Behnke (2014). „Learning depth-sensitive conditional random fields for semantic segmentation of RGB-D images.“ In: *ICRA*, pp. 6232–6237 (cit. on pp. 18, 24).
- Neverova, Natalia, Pauline Luc, Camille Couprie, Jakob Verbeek, and Yann LeCun (2017). „Predicting deeper into the future of semantic segmentation.“ In: *arXiv:1703.07684v2 [cs.CV]* (cit. on p. 139).
- Nowozin, Sebastian, Peter Gehler, and Christoph H Lampert (2010). „On parameter learning in CRF-based approaches to object class image segmentation.“ In: *ECCV*, pp. 98–111 (cit. on pp. 14, 15, 21).
- Nowozin, Sebastian and Christoph H Lampert (2011). „Structured learning and prediction in computer vision.“ In: *Foundations and Trends in Computer Graphics and Vision* 6.3-4, pp. 185–365 (cit. on pp. xvi, 15, 22, 38–40, 124).
- Nuss, Dominik, Ting Yuan, Gunther Krehl, Manuel Stuebler, Stephan Reuter, and Klaus Dietmayer (2015). „Fusion of laser and radar sensor data with a sequential Monte Carlo Bayesian occupancy filter.“ In: *IV Symposium*, pp. 1074–1081 (cit. on p. 111).
- OICA (2016). *2015 Production Statistics*. URL: <http://www.oica.net/category/production-statistics/2015-statistics/> (visited on 12/21/2016) (cit. on p. 6).
- Pan, Jiyan and Takeo Kanade (2013). „Coherent object detection with 3D geometric context from a single image.“ In: *ICCV*, pp. 2576–2583 (cit. on p. 18).
- Paszke, Adam, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello (2016). „ENet: A deep neural network architecture for

- real-time semantic segmentation." In: *arXiv:1606.02147v1 [cs.CV]* (cit. on pp. 17, 84, 85).
- Pfeiffer, David (2011). „The Stixel world: A compact medium-level representation for efficiently modeling dynamic three-dimensional environments." PhD thesis. Humboldt-Universität zu Berlin (cit. on pp. 109, 110, 119, 122, 123).
- Pfeiffer, David and Uwe Franke (2011a). „Modeling dynamic 3D environments by means of the Stixel world." In: *ITS Magazine* 3.3, pp. 24–36 (cit. on p. 108).
- (2011b). „Towards a global optimal multi-layer Stixel representation of dense 3D data." In: *BMVC*, pp. 51.1–51.12 (cit. on pp. iv, vi, 10, 11, 27, 29, 108, 110, 112, 113).
- Pfeiffer, David, Stefan K Gehrig, and Nicolai Schneider (2013). „Exploiting the power of stereo confidences." In: *CVPR*, pp. 297–304 (cit. on pp. 59, 120, 124).
- Pinheiro, Pedro O and Ronan Collobert (2014). „Recurrent convolutional neural networks for scene labeling." In: *ICML*, pp. 82–90 (cit. on p. 16).
- Plath, Nils, Marc Toussaint, and Shinichi Nakajima (2009). „Multi-class image segmentation using conditional random fields and global classification." In: *ICML*, pp. 817–824 (cit. on pp. 14, 15, 21).
- Platt, John C (1999). „Probabilistic outputs for support vector machines and comparisons to regular likelihood methods." In: *Advances in Large Margin Classifiers*. Ed. by A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans. MIT Press (cit. on p. 25).
- Pohlen, Tobias, Alexander Hermans, Markus Mathias, and Bastian Leibe (2017). „Full-resolution residual networks for semantic segmentation in street scenes." In: *CVPR*, to appear (cit. on pp. 16, 84, 85).
- Razavian, Ali Sharif, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson (2014). „CNN features off-the-shelf: An astounding baseline for recognition." In: *CVPR*, pp. 806–813 (cit. on p. 16).
- Redmon, Joseph, Santosh Divvala, Ross B Girshick, and Ali Farhadi (2016). „You only look once: Unified, real-time object detection." In: *CVPR*, pp. 779–788 (cit. on p. 125).
- Ren, Shaoqing, Kaiming He, Ross B Girshick, and Jian Sun (2015). „Faster R-CNN: Towards real-time object detection with region proposal networks." In: *NIPS*, pp. 91–99 (cit. on p. 49).
- Ren, Xiaofeng, Liefeng Bo, and Dieter Fox (2012). „RGB-(D) scene labeling: Features and algorithms." In: *CVPR*, pp. 2759–2766 (cit. on pp. 18, 21).
- Reynolds, Jordan and Kevin P Murphy (2007). „Figure-ground segmentation using a hierarchical conditional random field." In: *CRV*, pp. 175–182 (cit. on pp. 14, 15, 21).

- Richter, Stephan R, Vibhav Vineet, Stefan Roth, and Vladlen Koltun (2016). „Playing for data: Ground truth from computer games.“ In: *ECCV*, pp. 102–118 (cit. on p. 72).
- Rieken, Jens, Richard Matthaei, and Markus Maurer (2015). „Benefits of using explicit ground-plane information for grid-based urban environment modeling.“ In: *Fusion*, pp. 2049–2056 (cit. on p. 110).
- Riemenschneider, Hayko, András Bódis-Szomorú, Julien Weisenberg, and Luc Van Gool (2014). „Learning where to classify in multi-view semantic segmentation.“ In: *ECCV*, pp. 516–532 (cit. on pp. 18, 68).
- Ros, German, Sebastian Ramos, Manuel Granados, David Vazquez, and Antonio M Lopez (2015). „Vision-based offline-online perception paradigm for autonomous driving.“ In: *WACV*, pp. 231–238 (cit. on pp. 42, 57, 68, 69, 98, 99, 131).
- Ros, German, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez (2016). „The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes.“ In: *CVPR*, pp. 3234–3243 (cit. on p. 72).
- Russakovsky, Olga et al. (2015). „ImageNet large scale visual recognition challenge.“ In: *IJCV* 115.3, pp. 211–252 (cit. on pp. xv, 3, 4, 20, 57, 59, 68, 75, 84, 85, 94, 101).
- Russell, Bryan C, Antonio Torralba, Kevin P Murphy, and William T Freeman (2008). „LabelMe: A database and web-based tool for image annotation.“ In: *IJCV* 77.1-3, pp. 157–173 (cit. on p. 63).
- SAE International (2014). *Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems (J3016)*. Tech. rep. On-Road Automated Driving (Orad) Committee (cit. on p. 7).
- Said, Carolyn (1990). „DYCAM Model 1: The first portable digital still camera.“ In: *MacWeek* 4.35, p. 34 (cit. on p. 1).
- Sanberg, Willem P, Gijs Dubbelman, and Peter H N de With (2014). „Extending the Stixel world with online self-supervised color modeling for road-versus-obstacle segmentation.“ In: *ITSC*, pp. 1400–1407 (cit. on p. 110).
- Scharwächter, Timo, Markus Enzweiler, Uwe Franke, and Stefan Roth (2013). „Efficient multi-cue scene segmentation.“ In: *GCPR*, pp. 435–445 (cit. on pp. xiv, 14, 17, 20, 25, 42, 52, 58, 66, 68).
- (2014a). „Stixmantics: A medium-level model for real-time semantic scene understanding.“ In: *ECCV*, pp. 533–548 (cit. on pp. 20–22, 32, 40, 45–47, 49, 52, 53, 57).
- Scharwächter, Timo and Uwe Franke (2015). „Low-level fusion of color, texture and depth for robust road scene understanding.“ In: *IV Symposium*, pp. 599–604 (cit. on pp. 24, 25, 29, 109).
- Scharwächter, Timo, Manuela Schuler, and Uwe Franke (2014b). „Visual guard rail detection for advanced highway assistance systems.“ In: *IV Symposium*, pp. 900–905 (cit. on pp. 18, 125).

- Schneider, Johannes, Cyrill Stachniss, and Wolfgang Förstner (2016). „On the accuracy of dense fisheye stereo.“ In: *IEEE Robotics and Automation Letters* 1.1, pp. 227–234 (cit. on p. 32).
- Schneider, Lukas, Marius Cordts, Timo Rehfeld, David Pfeiffer, Markus Enzweiler, Uwe Franke, Marc Pollefeys, and Stefan Roth (2016). „Semantic Stixels: Depth is not enough.“ In: *IV Symposium*, pp. 110–117 (cit. on pp. 110, 123, 129).
- Schwing, Alexander G and Raquel Urtasun (2015). „Fully connected deep structured networks.“ In: *arXiv:1503.02351v1 [cs.CV]* (cit. on pp. 16, 75).
- Sengupta, Sunando, Eric Greveson, Ali Shahrokni, and Philip H S Torr (2013). „Urban 3D semantic modelling using stereo vision.“ In: *ICRA*, pp. 580–585 (cit. on pp. 18, 20, 42, 68, 69, 98, 99, 131).
- Sengupta, Sunando, Paul Sturgess, L’ubor Ladický, and Philip H S Torr (2012). „Automatic dense visual semantic mapping from street-level imagery.“ In: *IROS*, pp. 857–862 (cit. on p. 68).
- Shafaei, Alireza, James J Little, and Mark Schmidt (2016). „Play and learn: Using video games to train computer vision models.“ In: *BMVC* (cit. on p. 72).
- Sharma, Abhishek, Oncel Tuzel, and David W Jacobs (2015). „Deep hierarchical parsing for semantic segmentation.“ In: *CVPR*, pp. 530–538 (cit. on pp. 16, 45–47, 49).
- Shelhamer, Evan, Jonathan Long, and Trevor Darrell (2017). „Fully convolutional networks for semantic segmentation.“ In: *Trans. PAMI* 39.4, pp. 640–651 (cit. on pp. xv, 15, 16, 57, 66, 74, 75, 80, 95, 98, 129).
- Shotton, Jamie, Matthew Johnson, and Roberto Cipolla (2008). „Semantic texton forests for image categorization and segmentation.“ In: *CVPR*, pp. 1–8 (cit. on pp. 23–25, 45–50, 52, 53).
- Sigurdsson, Gunnar A., Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta (2016). „Hollywood in homes: Crowdsourcing data collection for activity understanding.“ In: *ECCV*, pp. 510–526 (cit. on p. 4).
- Silberman, Nathan, Derek Hoiem, Pushmeet Kohli, and Rob Fergus (2012). „Indoor segmentation and support inference from RGBD images.“ In: *ECCV*, pp. 746–760 (cit. on pp. xvi, 18, 20–22, 34, 46).
- Simonyan, Karen and Andrew Zisserman (2015). „Very deep convolutional networks for large-scale image recognition.“ In: *ICLR* (cit. on pp. xvii, 16, 75, 84, 85, 95, 96).
- Song, Shuran, Samuel P Lichtenberg, and Jianxiong Xiao (2015). „SUN RGB-D: A RGB-D scene understanding benchmark suite.“ In: *CVPR*, pp. 567–576 (cit. on pp. 6, 18, 68).
- Sturgess, Paul, Karteek Alahari, L’ubor Ladický, and Philip H S Torr (2009). „Combining appearance and structure from motion features for road scene understanding.“ In: *BMVC*, pp. 62.1–62.11 (cit. on p. 18).

- Szegedy, Christian, Sergey Ioffe, and Vincent Vanhoucke (2016a). „Inception-v4, Inception-ResNet and the impact of residual connections on learning.“ In: *ICLR Workshops* (cit. on p. 85).
- Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich (2015). „Going deeper with convolutions.“ In: *CVPR*, pp. 1–9 (cit. on pp. 85, 95–97, 129).
- Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna (2016b). „Rethinking the inception architecture for computer vision.“ In: *CVPR*, pp. 2818–2826 (cit. on p. 96).
- Thrun, Sebastian (2002). „Robotic mapping: A survey.“ In: *Exploring artificial intelligence in the new millennium*. Ed. by G Lakemeyer and B Nebel. Morgan Kaufmann Publishers Inc., pp. 1–35 (cit. on p. 111).
- Tighe, Joseph and Svetlana Lazebnik (2013). „Superparsing: Scalable nonparametric image parsing with superpixels.“ In: *IJCV* 101.2, pp. 329–349 (cit. on pp. 14, 15).
- Tomasi, Carlo and Takeo Kanade (1991). *Detection and tracking of point features*. Tech. rep. CMU-CS-91-132. Carnegie Mellon University (cit. on pp. xv, 41).
- Trembl, Michael et al. (2016). „Speeding up semantic segmentation for autonomous driving.“ In: *NIPS Workshop: MLITS* (cit. on pp. 17, 85).
- Uijlings, Jasper R R, Koen E A Van de Sande, Theo Gevers, and Arnold W M Smeulders (2013). „Selective search for object recognition.“ In: *IJCV* 104.2, pp. 154–171 (cit. on p. 111).
- Valentin, Julien P C, Sunando Sengupta, Jonathan Warrell, Ali Shahrokni, and Philip H S Torr (2013). „Mesh based semantic modelling for indoor and outdoor scenes.“ In: *CVPR*, pp. 2067–2074 (cit. on p. 18).
- Vedaldi, Andrea and Stefano Soatto (2008). „Quick shift and kernel methods for mode seeking.“ In: *ECCV*, pp. 705–718 (cit. on p. 14).
- Veksler, Olga, Yuri Boykov, and Paria Mehrani (2010). „Superpixels and supervoxels in an energy optimization framework.“ In: *ECCV*, pp. 211–224 (cit. on pp. xvi, 27, 29).
- Vineet, Vibhav et al. (2015). „Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction.“ In: *ICRA*, pp. 75–82 (cit. on pp. 98, 99).
- Viola, Paul and Michael J Jones (2004). „Robust real-time face detection.“ In: *IJCV* 57.2, pp. 137–154 (cit. on pp. 41, 125).
- Wang, Panqu, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell (2017). „Understanding convolution for semantic segmentation.“ In: *arXiv:1702.08502v1 [cs.CV]* (cit. on pp. 16, 85, 92).
- WHO (2015). *Global status report on road safety*. Tech. rep. World Health Organization (cit. on p. 6).

- Wojek, Christian, Stefan Walk, Stefan Roth, Konrad Schindler, and Bernt Schiele (2013). „Monocular visual scene understanding: Understanding multi-object traffic scenes.“ In: *Trans. PAMI* 35.4, pp. 882–897 (cit. on p. 18).
- Wu, Jianxin (2010). „A fast dual method for HIK SVM learning.“ In: *ECCV*, pp. 552–565 (cit. on pp. xv, 25, 33).
- Wu, Zifeng, Chunhua Shen, and Anton Van den Hengel (2016). „Wider or deeper: Revisiting the ResNet model for visual recognition.“ In: *arXiv:1611.10080v1 [cs.CV]* (cit. on pp. 16, 84, 85, 92, 94, 105).
- Xiao, Jianxiong, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva (2016). „SUN Database: Exploring a large collection of scene categories.“ In: *IJCV* 119.1, pp. 3–22 (cit. on pp. xvi, 3).
- Xie, Jun, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger (2016). „Semantic instance annotation of urban scenes by 3D to 2D label transfer.“ In: *CVPR*, pp. 3688–3697 (cit. on pp. 61, 68, 71).
- Xu, Chenliang, Spencer Whitt, and Jason J Corso (2013). „Flattening supervoxel hierarchies by the uniform entropy slice.“ In: *ICCV*, pp. 2240–2247 (cit. on pp. 42, 68, 69).
- Xu, Philippe, Franck Davoine, Jean-Baptiste Bordes, Huijing Zhao, and Thierry Denoeux (2013). „Information fusion on oversegmented images: an application for urban scene understanding.“ In: *MVA*, pp. 189–193 (cit. on p. 131).
- Yang, Yi, Sam Hallman, Deva Ramanan, and Charless Fowlkes (2010). „Layered object detection for multi-class segmentation.“ In: *CVPR*, pp. 3113–3120 (cit. on p. 14).
- Yao, Jian, Sanja Fidler, and Raquel Urtasun (2012). „Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation.“ In: *CVPR*, pp. 702–709 (cit. on pp. 14, 15, 20, 21).
- Yu, Fisher and Vladlen Koltun (2016). „Multi-scale context aggregation by dilated convolutions.“ In: *ICLR* (cit. on pp. 16, 85, 98, 105).
- Zhang, Chenxi, Liang Wang, and Ruigang Yang (2010). „Semantic segmentation of urban scenes using dense depth maps.“ In: *ECCV*, pp. 708–721 (cit. on p. 18).
- Zhang, Jian, Chen Kan, Alexander G Schwing, and Raquel Urtasun (2013). „Estimating the 3D layout of indoor scenes and its clutter from depth sensors.“ In: *ICCV*, pp. 1273–1280 (cit. on p. 18).
- Zhang, Richard, Stefan A Candra, Kai Vetter, and Avidesh Zakhori (2015). „Sensor fusion for semantic segmentation of urban scenes.“ In: *ICRA*, pp. 1850–1857 (cit. on pp. 18, 68, 69, 131).
- Zhang, Shanshan, Rodrigo Benenson, and Bernt Schiele (2017). „CityPersons: A diverse dataset for pedestrian detection.“ In: *arXiv:1702.05693v1 [cs.CV]* (cit. on p. 140).



- Zhao, Hengshuang, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia (2017). „Pyramid scene parsing network.“ In: *CVPR*, to appear (cit. on pp. [16](#), [85](#), [95](#), [97](#), [98](#)).
- Zheng, Shuai, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H S Torr (2015). „Conditional random fields as recurrent neural networks.“ In: *ICCV*, pp. 1529–1537 (cit. on pp. [16](#), [75](#)).
- Zheng, Ying, Steve Gu, and Carlo Tomasi (2012). „Fast tiered labeling with topological priors.“ In: *ECCV*, pp. 587–601 (cit. on p. [17](#)).
- Zhou, Bolei, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva (2016). „Places: An image database for deep scene understanding.“ In: *arXiv:1610.02055v1 [cs.CV]* (cit. on pp. [3](#), [85](#)).



## VITA

---

### MARIUS CORDTS

Date of birth: June 7<sup>th</sup>, 1988

Place of birth: Aachen, Germany

### EDUCATION

- 2013-2017 Technische Universität Darmstadt, Germany  
Ph.D. in Computer Science
- 2010-2013 RWTH Aachen University, Germany  
M.Sc. in Electrical Engineering, Information Technology  
and Computer Engineering
- 2007-2010 RWTH Aachen University, Germany  
B.Sc. in Electrical Engineering, Information Technology  
and Computer Engineering
- 1998-2007 Bischöfliches Gymnasium St. Ursula, Geilenkirchen, Ger-  
many  
Allgemeine Hochschulreife

### PROFESSIONAL EXPERIENCE

- Since 2015 Daimler AG, Böblingen, Germany  
Software Engineer: Environment Perception
- 2013-2015 Daimler AG, Böblingen, Germany  
Ph.D. Student: Environment Perception
- 2012 Siemens Corporate Research, Princeton, NJ, USA  
Intern: Imaging and Computer Vision
- 2011-2012 Rohde & Schwarz GmbH & Co. KG, Munich, Germany  
Intern: R&D RF Design Mobile Radio Testers

### HONORS

- 2014 Friedrich-Wilhelm-Prize
- 2014 SEW Eurodrive Foundation Graduate Award
- 2011-2013 Siemens Masters Program
- 2009-2013 Studienstiftung des Deutschen Volkes
- 2011 Henry-Ford II Studies Prize
- 2009-2011 NRW Scholarship Program
- 2010 Rohde & Schwarz Best Bachelor Prize
- 2009 Philips Bachelor Prize



## PUBLICATIONS

---

The work conducted in the context of this dissertation has led to the following publications:

**Marius Cordts**, Lukas Schneider, Markus Enzweiler, Uwe Franke, Stefan Roth. “Object-level Priors for Stixel Generation.” In: *German Conference on Pattern Recognition (GCPR)* [oral], 2014.

**Marius Cordts**, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele. “The Cityscapes Dataset.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR): Workshop on The Future of Datasets in Vision*, 2015.

Lukas Schneider\*, **Marius Cordts**\*, Timo Rehfeld, David Pfeiffer, Markus Enzweiler, Uwe Franke, Marc Pollefeys, Stefan Roth. “Semantic Stixels: Depth is Not Enough.” In: *IEEE Intelligent Vehicles Symposium (IV)* [oral, best paper award], 2016.

**Marius Cordts**, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele. “The Cityscapes Dataset for Semantic Urban Scene Understanding.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [spotlight], 2016.

Jonas Uhrig, **Marius Cordts**, Uwe Franke, Thomas Brox. “Pixel-level Encoding and Depth Layering for Instance-level Semantic Labeling.” In: *German Conference on Pattern Recognition (GCPR)* [oral, honorable mention award], 2016.

**Marius Cordts**\*, Timo Rehfeld\*, Markus Enzweiler, Uwe Franke, Stefan Roth. “Tree-Structured Models for Efficient Multi-Cue Scene Labeling.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 39.7, 2017.

**Marius Cordts**\*, Timo Rehfeld\*, Lukas Schneider\*, Markus Enzweiler, Stefan Roth, Marc Pollefeys, Uwe Franke. “The Stixel World: A Medium-Level Representation of Traffic Scenes.” In: *Image and Vision Computing (IVC)* 2017.

---

\* Authors contributed equally



## MEDIA COVERAGE

---

The work conducted in the context of this dissertation appeared in media as follows:

Press release by Koert Groeneveld and Katharina Becker, October 7<sup>th</sup>, 2015 on UR:BAN closing presentation, Düsseldorf, Germany; poster stand and vehicle demonstration by Marius Cordts

<http://media.daimler.com/marsMediaSite/ko/en/9919727>

Nvidia Keynote CES 2016, Las Vegas, NV, USA, January 4<sup>th</sup>, 2016. Talk by Jen-Hsun Huang and Michael Houston referring to the Cityscapes dataset

<https://youtu.be/HJ58dbd5g8g>

Mercedes-Benz Detroit NAIAS New Year's Reception 2016, Detroit, MI, USA, January 10<sup>th</sup>, 2016. Talk by Dr. Dieter Zetsche showing pixel-level semantic labeling results obtained by Marius Cordts

<http://media.daimler.com/marsMediaSite/ko/en/9920198>

[https://youtu.be/2yE\\_35yjMyw](https://youtu.be/2yE_35yjMyw)

Article by Dirk Gulde, December 28<sup>th</sup>, 2016 showing pixel-level semantic labeling results obtained by Marius Cordts

<http://www.auto-motor-und-sport.de/news/>

[autonomes-fahren-kuenstliche-intelligenz-11624000.html](http://www.auto-motor-und-sport.de/news/autonomes-fahren-kuenstliche-intelligenz-11624000.html)

Article by Sven Hansen, January 5<sup>th</sup>, 2017 on a talk by Marius Cordts on artificial intelligence at CES 2017, Las Vegas, NV, USA

<https://heise.de/-3588654>

Article by Roberto Baldwin, February 1<sup>st</sup>, 2017 referring to the Semantic Stixel World

<https://www.engadget.com/2017/01/02/>

[inside-mercedes-silicon-valley-research-center](https://www.engadget.com/2017/01/02/inside-mercedes-silicon-valley-research-center)

Article by Christoph Stockburger, February 4<sup>th</sup>, 2017 citing from an interview with Marius Cordts

<http://www.spiegel.de/auto/aktuell/a-1132759.html>